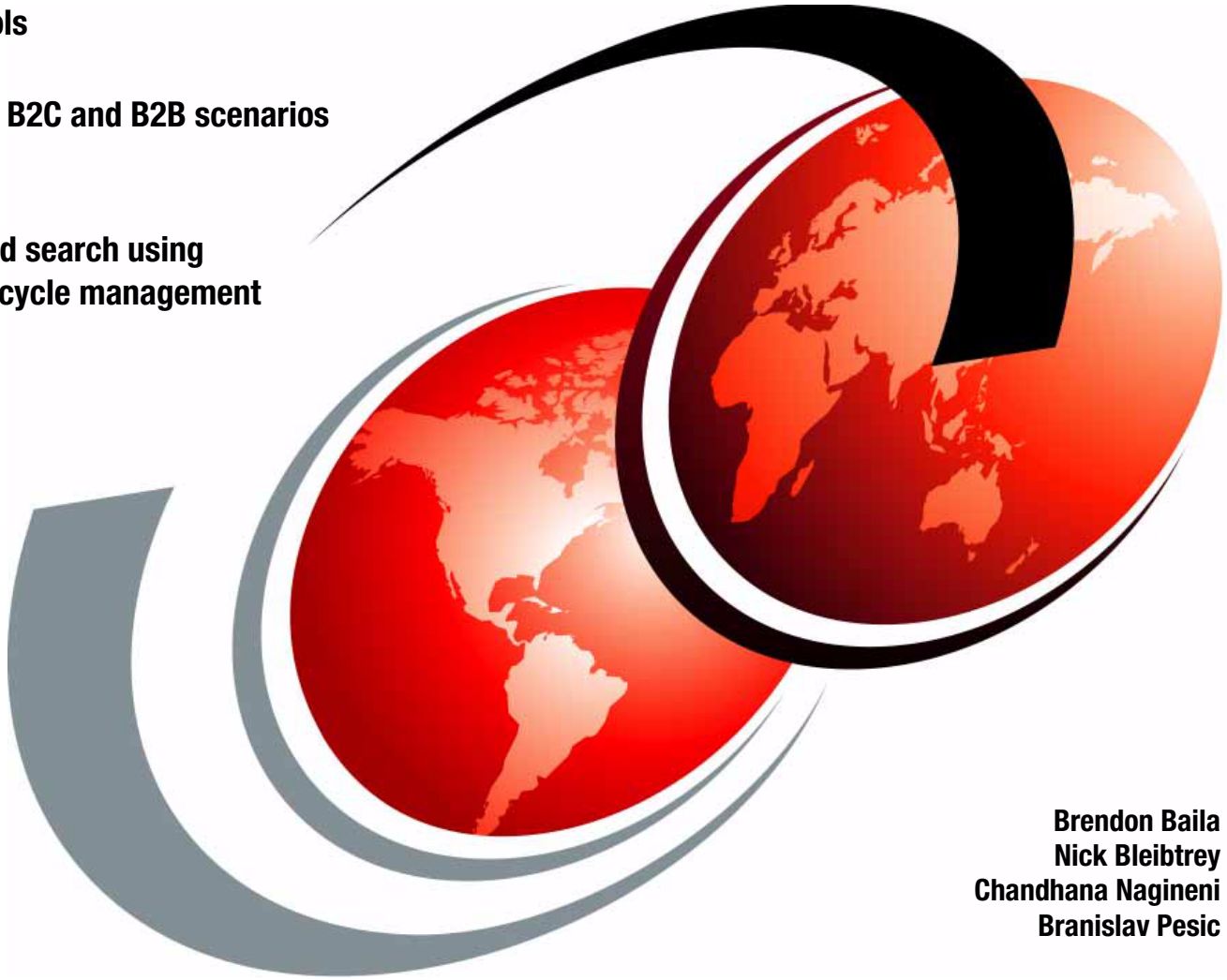
IBM® WebSphere®

WebSphere Commerce V7.0 Feature Pack 2 Search Solution Overview and Deployment

- Search results by using a new set of search tools
- Work with B2C and B2B scenarios
- Understand search using index life-cycle management



Brendon Baila
Nick Bleibtrey
Chandhana Nagineni
Branislav Pesic

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebSphere Commerce V7.0 Feature Pack 2 Search Solution Overview and Deployment

July 2011

SG24-7950-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (July 2011)

This edition applies to WebSphere Application Server V7.0, WebSphere Commerce V7.0, and WebSphere Commerce V7.0 Feature Pack 2.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DB2 Universal Database™	Passport Advantage®	Redpaper™
DB2®	Rational®	Redbooks (logo) ®
IBM®	Redbooks®	WebSphere®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication focuses on the enhanced search capabilities of the newest release of WebSphere Commerce V7.0 Feature Pack 2. We divided this book into three parts to highlight search business-to-consumer (B2C) and search business-to-business (B2B) scenarios and a conceptual overview of the search solution.

This book can help you to enable the search features of WebSphere Commerce V7.0 Feature Pack 2 and set up the WebSphere Commerce search environment, experience the design and management of Solr indexes, and experience several B2C scenarios using the Madisons Feature Pack 2 (FEP2) store. Madisons store is an IBM product within WebSphere Commerce.

This book gives you a broad understanding of the WebSphere Commerce integrated search solution that provides control over search results through a new set of search management tools:

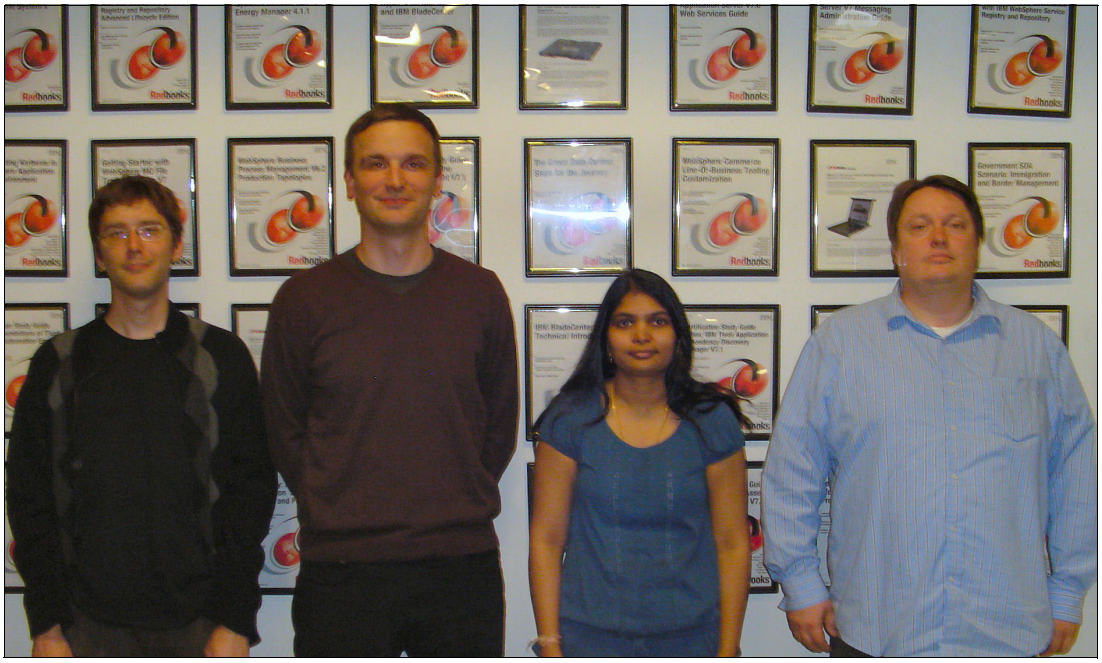
- Search term association in the Catalogs tool
- Marketing web activity tool
- Marketing dialog activity tool
- Search activity tool

This book describes search index life-cycle management and examines each area, such as index building, synchronization, configuration, and replication. This book describes the entitlement feature that is provided through the WebSphere Commerce search framework. We also discuss a B2B scenario and provide the implementation using catalog filters and contracts.

This book is designed for use by WebSphere Commerce developers, practitioners, and solution architects in various industries.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.



Authors from left to right, Brendon Baila, Branislav Pesic, Chandhana Nagineni, and Nick Bleibtrey

Brendon Baila is a Client Technical Resolution Specialist for WebSphere Commerce. He joined IBM (Toronto Lab) in 2009 after graduating from McMaster University in Computer Engineering.

Nick Bleibtrey is an experienced WebSphere® Commerce Architect and Developer with multiple successful client engagements. He currently focuses on the WebSphere Commerce client engagement manager and technical sales roles. He received his bachelors of science degree in Computer Science and Applied Mathematics from the State University of New York at Albany. He works for Sirius Computer Solutions (http://www.siriuscom.com/).

Chandhana Nagineni is a Solutions Architect at Miracle Software Systems, Inc. She holds a bachelor degree in Computer Science from JNT University in India. She has over nine years of experience working with the large retail, manufacturing, and healthcare companies. Her areas of expertise include aligning business strategies to IT strategy, developing enterprise architecture, business process management (BPM), and service-oriented architecture. She specializes in WebSphere Application Server, WebSphere Commerce server, WebSphere Portal Server, and Sterling Integrator. Her specialties in WebSphere Commerce server are building extended sites, B2B, B2C stores, and integration with enterprise resource planning (ERP) systems. She also works with pre-sales teams to build proof of concept (POC)/proof of technology (POT), return on investment (ROI)/total cost of ownership (TCO) analysis, and competitive analysis.

Branislav Pesic is a Consultant in Serbia. He works at the European branch of Ascendant Technology, a US-based IBM Premier Business Partner. He has six years of experience in Java™ application development and design, transactional services for mobile operators and m-Government and e-Government system integration. He holds a bachelor's degree in Computer Science from University of Belgrade. His areas of expertise include customizing WebSphere Commerce tools, developing and debugging WebSphere Commerce, and integrating WebSphere Commerce with external systems.

Thanks to the following people for their contributions to this project:

Tamikia Barrow and Shari Deiana
International Technical Support Organization, Rochester Center

Jim Reach, Client Technical Sales, IBM Sales & Distribution, Software Sales
IBM Atlanta

Rufus P. Credle Jr., Certified Consulting IT Specialist, ITSO GCS, IBM Sales and Distribution
IBM Raleigh

Michael Au
IBM Toronto

Stan Bliakhman, SOA Commerce Development
IBM Toronto

Daniel Dunn, Xiangxi Lin, Adrian Baijnauth, Keith Chan, and Zhao Yan
Toronto Lab Commerce Development
IBM Toronto

David La Gamba and Mark Barbara
Toronto Lab Commerce Support
IBM Toronto

David Ruocco
Toronto Lab Commerce ID team
IBM Toronto

Nicola Byrne
Ireland Commerce Support
IBM Ireland

Vedavyas Avula, Lead Architect, Miracle Software Systems, Inc.

Andrew Beyer, eCommerce Operations Manager at Fossil

Fatih Akgul, Technology Manager at Rosetta

Howard Norton, WebSphere Commerce Architect at Fossil

Matthew Zipay, WebSphere Commerce Solution Architect at CrossView

Stephanie Zaremba, WebSphere Commerce Program Manager at Sterling Jewelers

Kenn Busse, eCommerce Development Manager at Fossil

Craig Pasquale, Director of eCommerce Operations
Ascendant Technology US

Radoslav Didic, Chief Operating Officer
Ascendant Technology Europe

Vesna Stupar Jovanic, Administrative Assistant
Ascendant Technology Belgrade

Igor Stojkovic, Consultant
Ascendant Technology Belgrade

Nenad Taskovic, Consultant
Ascendant Technology Belgrade

Vukasin Nikodijevic, Consultant
Ascendant Technology Belgrade

Igor Vulovic, Consultant
Ascendant Technology Belgrade

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Part 1

Conceptual overview of the search solution

The WebSphere Commerce integrated search solution provides control over search results through a new set of search management tools. Chapter 3, “Search business rules” on page 11 discusses these search management tools:

- Search term association in the catalog tool for landing page, synonym, and replacements term management
- Marketing web activity tool to manage product recommendations, cross-sell, and up-sell based on user search terms
- Marketing dialog activity tool to manage triggers and actions for product recommendation-based user search terms
- Search activity tools to manage the business rules to rank, sort, and display the top search results

The first group of tools under the catalog tools are for general use by catalog managers. The second group of integrated search tools are part of the Management Center Marketing tool. Marketing managers use the Management Center Marketing tool to define business rules that influence the content and ranking of search results within the store. The combination of search term associations and search marketing rules determines the final search results that are displayed to the shopper. Chapter 3, “Search business rules” on page 11 covers multiple scenarios and the steps to implement them.

This part of the book also discusses search index life-cycle management and examines each area, such as index building, synchronization, configuration, and replication, in more detail in Chapter 4, “Search index life-cycle management” on page 93.

This part concludes with tips about troubleshooting search-related issues in Chapter 5, “Troubleshooting” on page 101. The troubleshooting information also covers the common issues that are encountered during search index life-cycle management, Solr deployment, and maintenance and the recommended solutions.

1

Introduction

In December 2010, IBM released WebSphere Commerce V7.0 Feature Pack 2 (FEP 2). This product update expands and extends the core web sales platform with a number of enhancements:

- Customer-centric features:
  - Integrated search
  - Social bridging
  - Co-shopping
- Business-to-business (B2B) price rules
- Cross-channel optimization through integration with Sterling Commerce
- Additional foundational leadership features:
  - Content version handling
  - Attribute dictionary
  - Catalog filter
  - Store management
  - Web feed integration

Primarily, this book focuses on search capabilities; however, we divided it into several parts to highlight Search business-to-consumer (B2C) and B2B scenarios. *Integrated search* discusses how to implement the new customer-centered search capabilities of the product. We describe social bridging in addition to how to expand the marketing presence of your e-Commerce website by using the features that are introduced in FEP 2: integrated search and social bridging.

2

Search solution architecture

WebSphere Commerce search provides enhanced search functionality in starter stores by providing enriched search engine capabilities, such as automatic search term suggestions and spelling correction. WebSphere Commerce search influences store search results by using search term associations and search-based merchandising rules.

WebSphere Commerce search provides the following key business benefits:

- Resides on the top of open architecture
- Contains a rich set of search functionality for shoppers in starter stores
- Provides integrated search management tooling for business users in the Management Center
- Lowers the total cost of deployment and ownership, because its functionality is included as a feature of WebSphere Commerce

WebSphere Commerce search runs as a search server for your starter store that helps deliver targeted search results, while enabling customers to find products more effectively. It provides advanced administration features, such as dynamic clustering, database integration, rich document handling, distributed search, and index replication.

Figure 2-1 on page 6 illustrates WebSphere Commerce search.

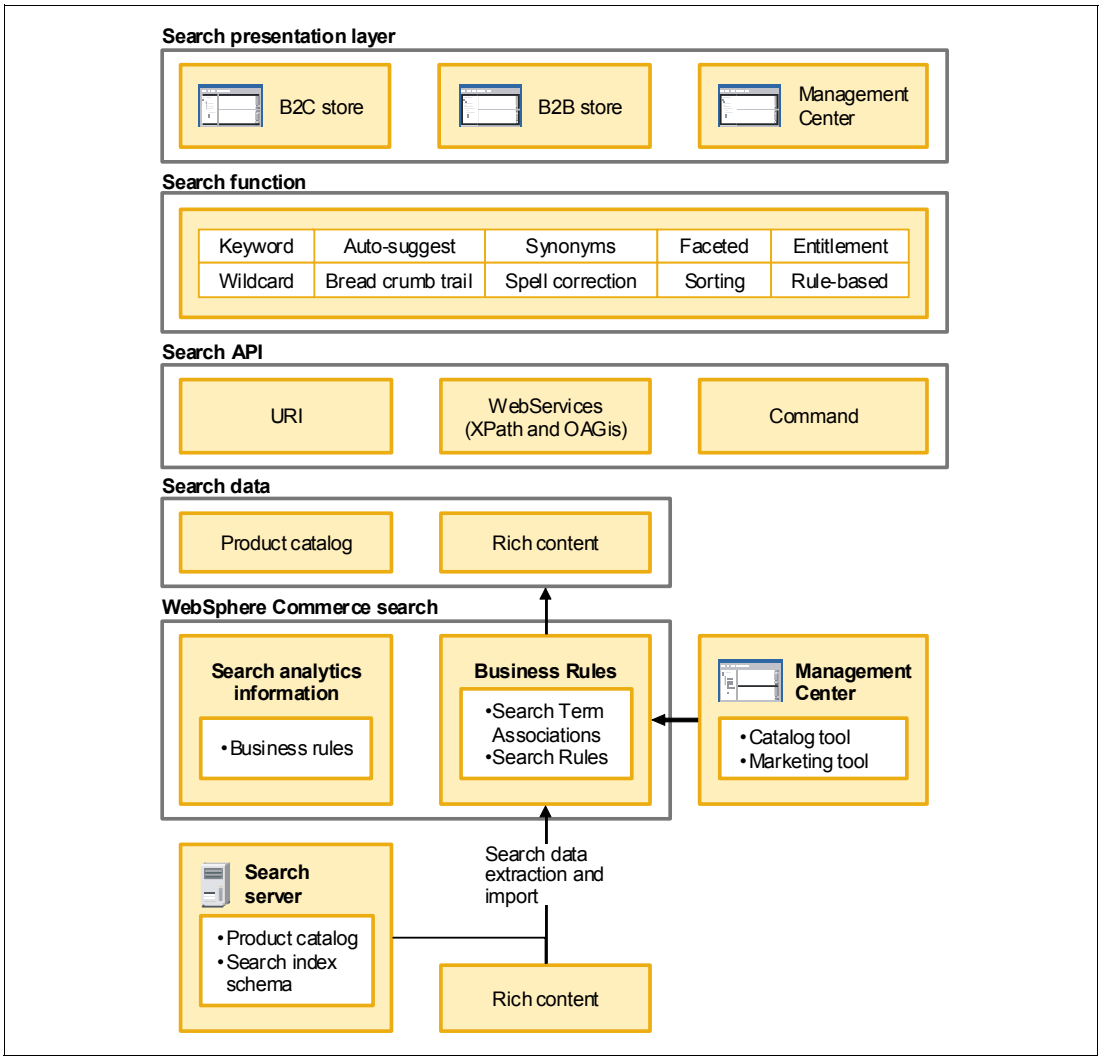


Figure 2-1 WebSphere Commerce search

WebSphere Commerce search has these characteristics:

- WebSphere Commerce search works with the search index to provide search results consisting of structured and unstructured content, and external data.
- Structured content is obtained from the store's product catalog using a product catalog adapter.
- Unstructured content is obtained as a document and file adapter and crawler.
- External content, such as product ratings from BazaarVoice, is obtained using an external data adapter and crawler.
- All content is merged and presented in starter store pages.

2.1 Understanding WebSphere Commerce search

WebSphere Commerce search is based around the search index, where an index is represented by a large flat table containing searchable fields that are optimized for search performance. The search index must be built before it can be used for any searches. The

search index is built with one or more *documents*, with documents containing fields. A *field* consists of a name, the content, and the metadata that tells WebSphere Commerce search how to handle the content. Typically, fields can contain boolean values, numbers, or strings. A field is flexible, enabling you to define your own type. Fields contain two important attributes: indexed and stored. If a field is indexed, it is searchable, sortable, and facetable. If an index is stored, its value is retrievable during a search.

WebSphere Commerce search uses a schema that is represented similarly to a database schema. The search schema defines the structure of the search index. Typically, the search configurations are performed in the search schema configuration file. This configuration file contains information about which fields your documents can contain, and how those fields must be processed when adding documents to the index, or when querying those fields.

The schema contains three major sections:

- Data types

  Enables you to define a list of field type declarations to use in your schema
- Fields

  Contains a list of the individual field declarations that you want to use in your documents
- Miscellaneous

  Contains miscellaneous information

WebSphere Commerce search uses HTTP requests to initiate indexing and searching. These requests are typically wrapped using the utilities that are provided with WebSphere Commerce search. During the indexing process, WebSphere Commerce search takes input documents as XML and builds an index using the information, and uses each field in the input document to build a searchable column. After the index is built, it can be queried using search commands. These search queries are HTTP requests, where if a match is found, it returns XML data as the search results.

2.2 Using WebSphere Commerce search to influence starter store search results

The following search tooling and features are available by default to influence search results in your starter store.

2.2.1 Search term associations

The product manager is responsible for creating and managing search term associations. *Search term associations* include synonyms and search term replacements, and linking search terms with landing pages.

The following features help customers to search the storefront:

- Synonyms increase the scope of search results by adding additional search terms to the customer's search submission.
- Replacement terms modify the potential search results by changing the search terms from the customer's search submission.
- Landing pages promote certain products or activities in the store by directing the customer to specific store pages that are based on the customer's search submission.

See 3.1, “Search Term Association tool” on page 12 for more information to help you improve storefront search results.

2.2.2 Search-based merchandising rules

Search rules are managed by marketing managers. They are used when customers search the storefront to deliver customized search results and ordering.

For example, you can create the following search rules using the Search Rule Builder:

- When a customer searches the store, change the relative order of the search result by favoring catalog entries that belong to a specific brand or category, or sort catalog entries by price.
- When a customer searches the store, promote specific catalog entries to the top of the search result.
- When a customer searches the store, sort the catalog entries by the manufacturer names.

See Chapter 3, “Search business rules” on page 11 for more information.

2.2.3 Integrating starter stores

You can deliver powerful search-based catalog browsing flows by enabling WebSphere Commerce search in starter stores. Starter store enhancements for the storefront, such as automatic search term suggestions, spelling correction, and search result highlighting, are provided.

See “Enabling the search-based navigation store function” on page 134 for more information about WebSphere Commerce search in starter stores.

2.2.4 Limitations

The following limitations exist in WebSphere Commerce search-enabled starter stores:

- Coremetrics is not supported by default, therefore analytics cannot be used to influence store behavior.
- You cannot preview the store changes that are made in workspaces by default, because the workspace schema is not indexed.
- If you consider integrating with third-party search solution functionality, you must consider the scope of the customization points that are listed in *Integrating WebSphere Commerce search with third-party search engines* at the following website:

  `http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.developer.doc/concepts/csdsearch.htm`
- The Madisons mobile starter store does not support WebSphere Commerce search.

For more information regarding WebSphere Commerce search, visit the following web link:

`http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.developer.doc/concepts/csdsearch.htm`

Part 2

Business-to-consumer (B2C) scenario

In this part of the book, we discuss how to enable the search features of WebSphere Commerce V7.0 Feature Pack 2 and set up the WebSphere Commerce search environment.

We discuss the design and management of Solr indexes, an overview of the IBM WebSphere Commerce V7.0 catalog index, and an example of how to customize the catalog index to handle inventory data.

We also describe the storefront, how to enable the search-based navigation for a storefront, and all the new features that are available as part of the predefined search when enabled. We implemented all the scenarios in this chapter in the Madison Feature Pack 2 (FEP2) business-to-consumer (B2C) store.

3

Search business rules

Search business rules are used to deliver custom results to the shopper when the shopper searches the store for specific keywords. This chapter describes the tools that are available in the Management Center of WebSphere Commerce that enable the marketing managers and catalog managers to create search rules.

In this chapter, we introduce the search rules, web activity, and dialog activity that are available in the Management Center. These tools provide the ability to create rules. This chapter discusses the following tools, which are used to complete the scenario, that are available in the Management Center:

- Search Term Association tool
- Search Marketing tool

This chapter also discusses the graphical editor called the *Search Rule Builder,* which is available in the Management Center. The Search Rule Builder is used to create search rules that are triggered by a customer submitting a search.

We implemented all the scenarios in this chapter in the Madison Feature Pack 2 (FEP2) business-to-consumer (B2C) store. We performed most of the search scenarios by entering the search term in the quick search tool in the header. The breadcrumb trail in the Search Results page displays the input search term that was used.

3.1 Search Term Association tool

The Search Term Association tool is a user interface that is defined in the Catalogs tool in Management Center. It allows business users to modify search queries and affect the search results. It is used to suggest similar, separate, or replacement products in the search results. The Search Term Association tool also is used to display a landing page as a search result instead of products.

The following association types are available:

- Synonyms (bidirectional)
- Replacements (unidirectional)
- Landing pages

The Search Term Association tool is only available if these conditions are true:

- Feature Pack 2 is installed and the store enhancements feature is enabled.
- The store's master catalog is indexed for WebSphere Commerce search.
- The search server and index structure are deployed and built.

Open search term associations

Follow these steps to open the search term associations:

1. Log on to Management Center:

   http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.management-center.doc/tasks/ttflogon.htm

2. Open **Catalogs** tool, as shown in Figure 3-1.

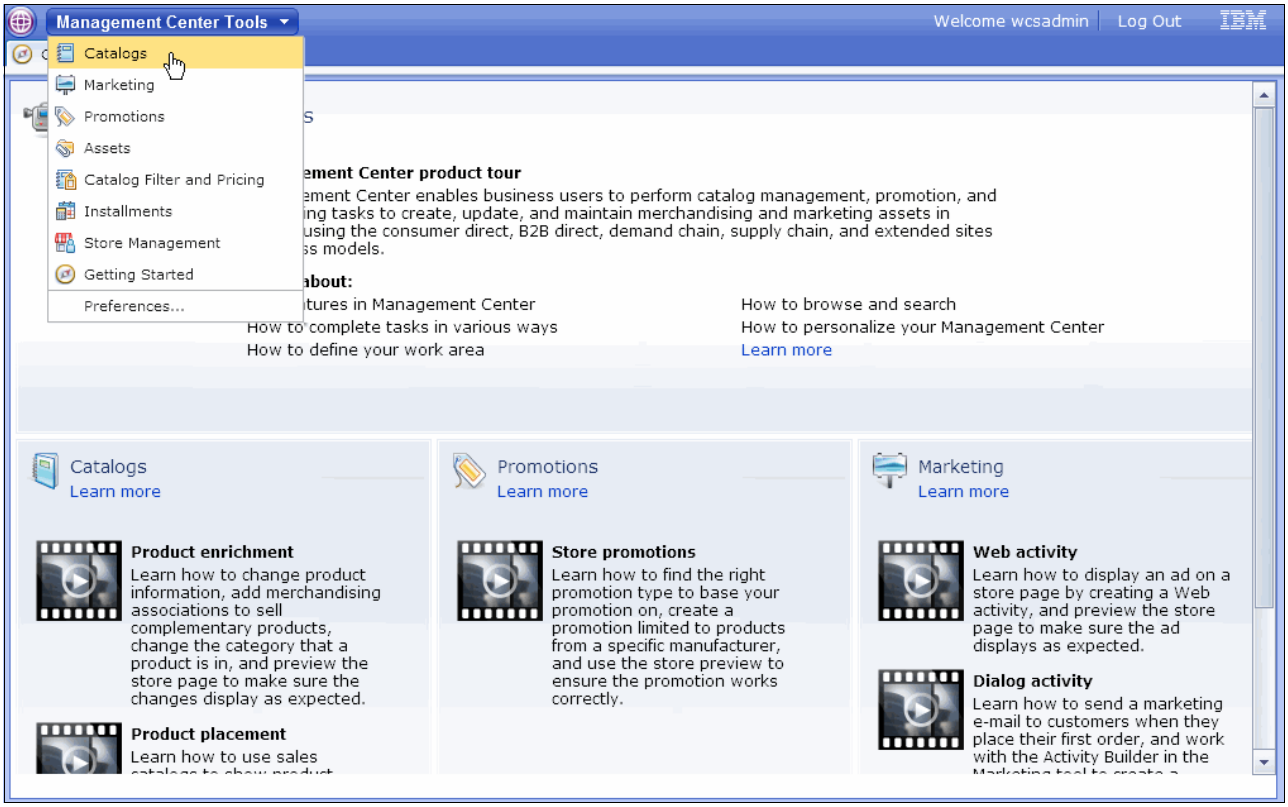


Figure 3-1 Open Catalogs tool

3. Select the store from the upper-right drop-down list, as shown in Figure 3-2.

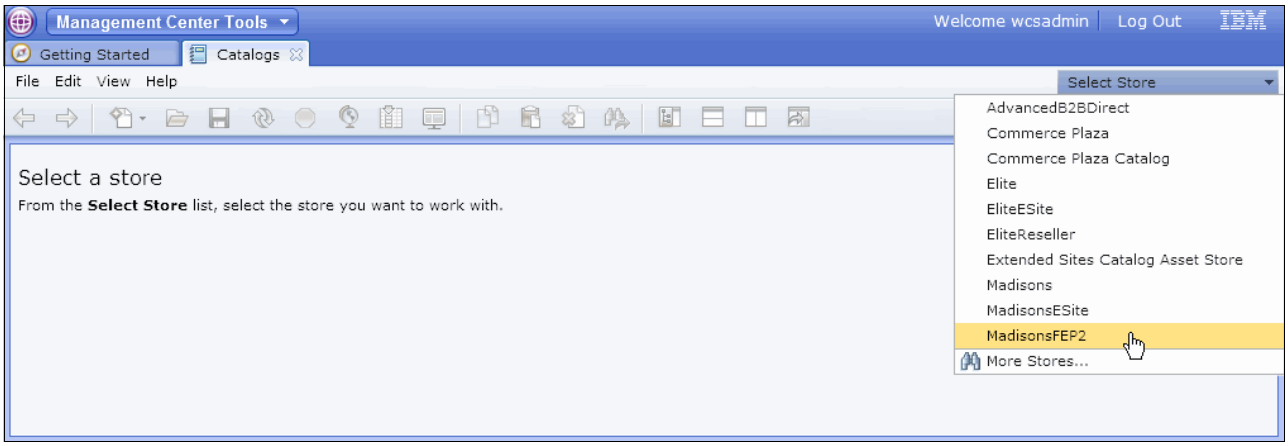


Figure 3-2 Select MadisonsFEP2 store

4. Click **Search Term Associations** in the explorer tree to open the Search Term Associations tool. You see a window similar to Figure 3-3.

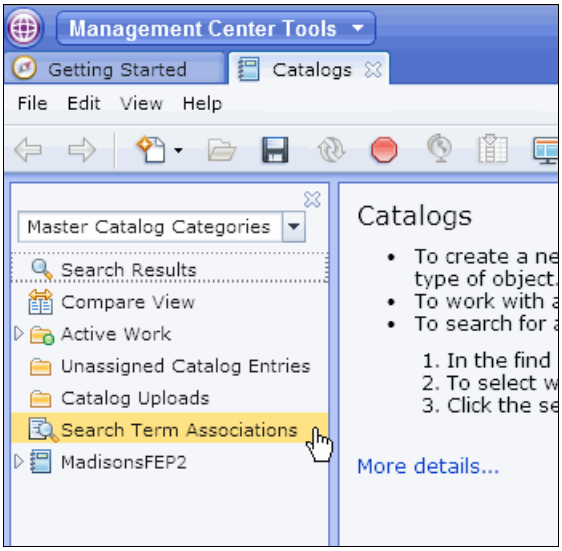


Figure 3-3 Open Search Term Associations tool

3.1.1 Synonyms

Synonyms are used to indicate the keywords to search for apart from the input search terms. The search results include the submitted search term, plus the search results for the additional defined synonyms. One or more synonyms can be defined for a search term. Synonyms are bidirectional: searching for any of the search terms expands to also search for the other synonym search terms.

In scenarios where you have to search for products having keyword B, which is similar to another product having keyword A, you use synonyms. In the current scenario, the business has a requirement to search also for all products having the keyword `jeans` whenever the shopper searches for the keyword `pants`. Before defining the synonyms, the search results for the search term `pants` are shown in Figure 3-4 on page 14 and the search results for the search term `jeans` are shown in Figure 3-5 on page 15.

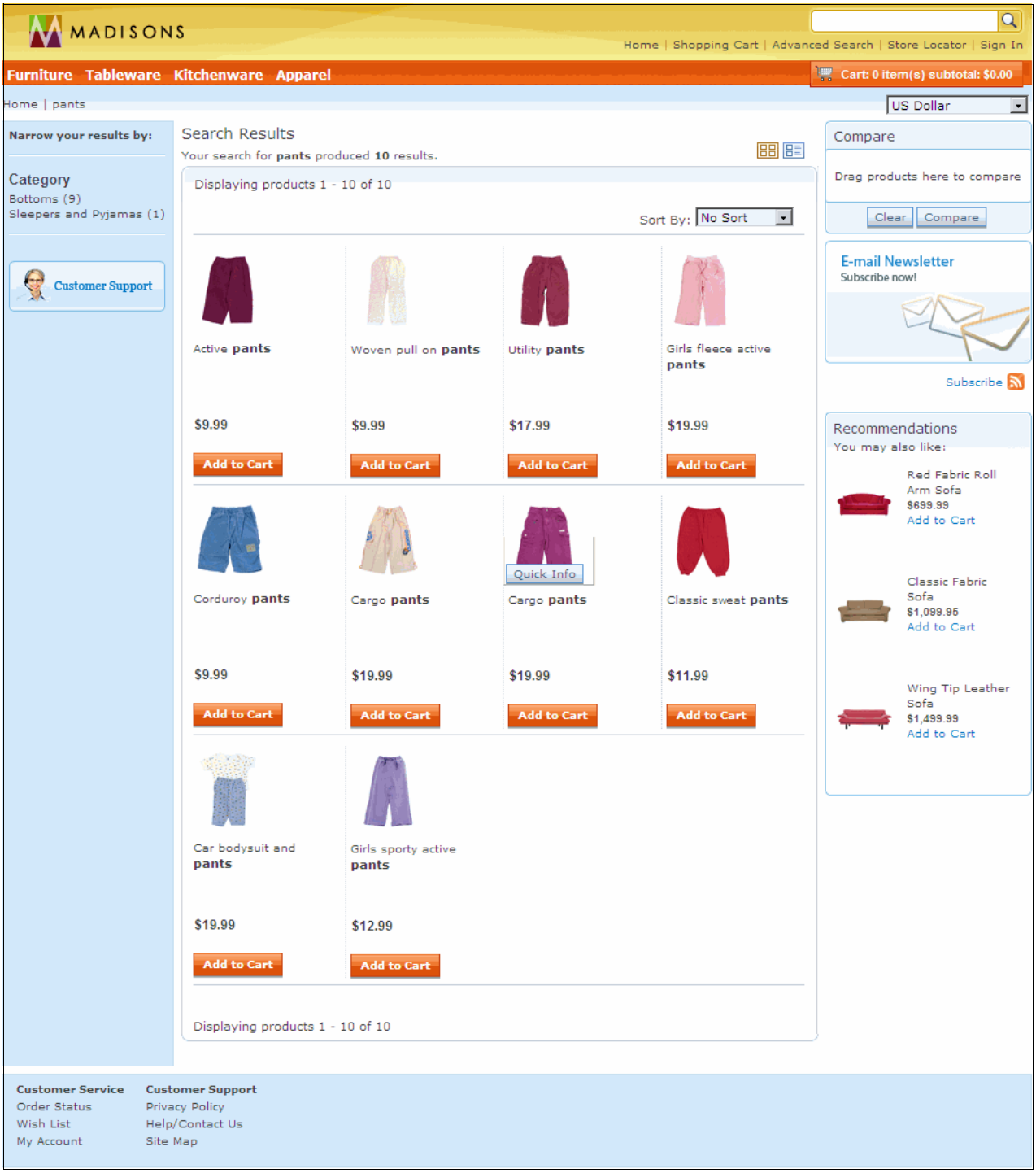


Figure 3-4 Search results for pants before adding the synonyms

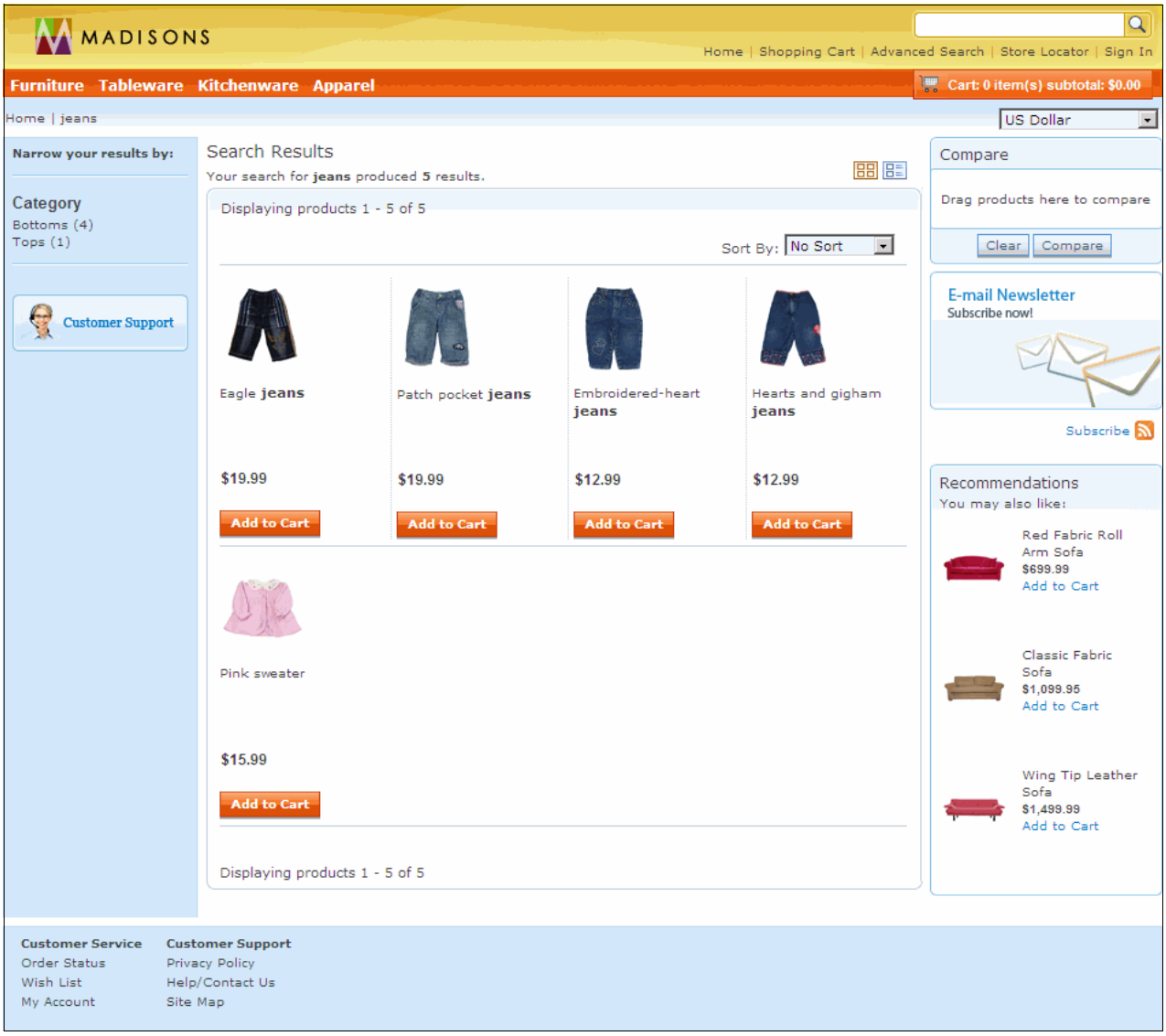


Figure 3-5 Search results for jeans before adding the synonyms

Complete these steps to add synonyms for the search term `pants`:

1. Follow the steps in “Open search term associations” on page 12.
2. Click the **Synonyms** tab if it is not opened by default.
3. Click **Create New Synonym**, as shown in Figure 3-6 on page 16.

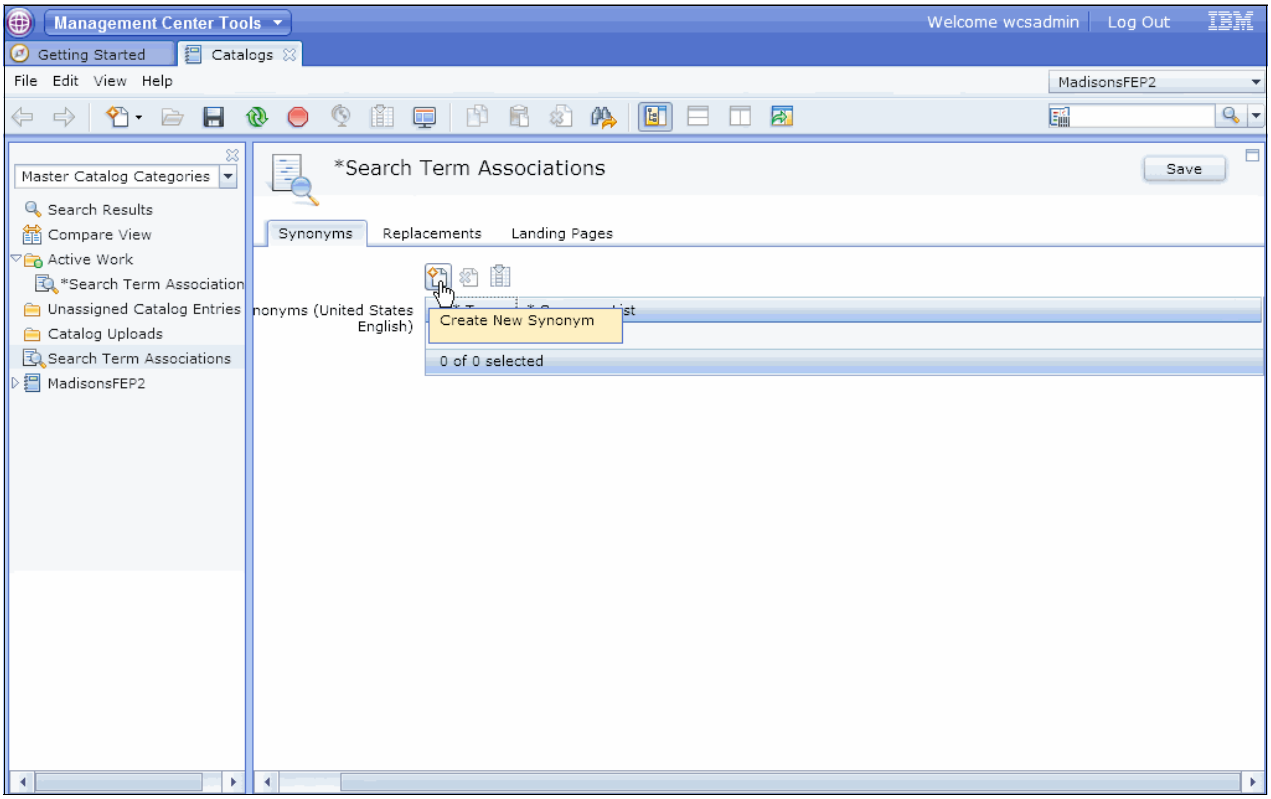


Figure 3-6 Create New Synonym

4. Enter the values `pants, jeans`, as shown in Figure 3-7.

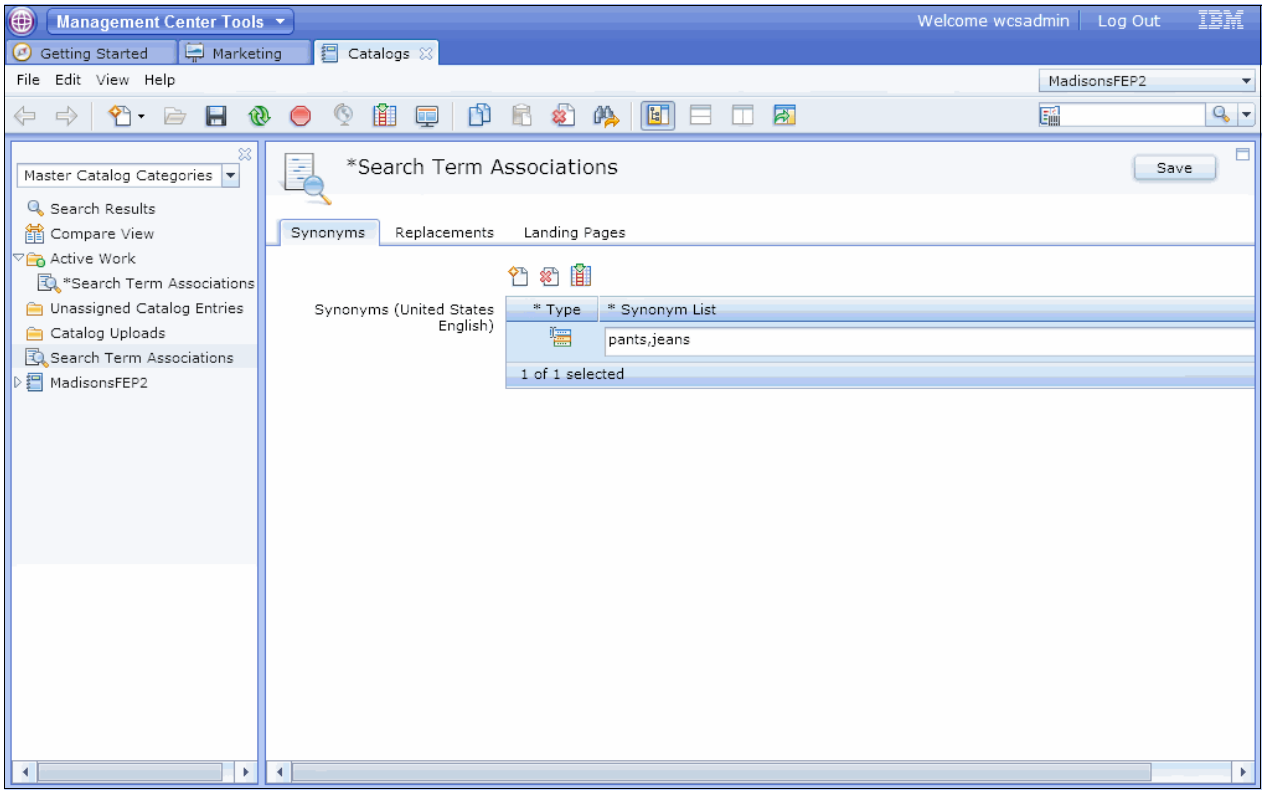


Figure 3-7 Synonyms for pants added

5. Click **Save**.

The search results for `pants` in the storefront now contain the results for the synonym term `jeans` also, as shown in Figure 3-8.

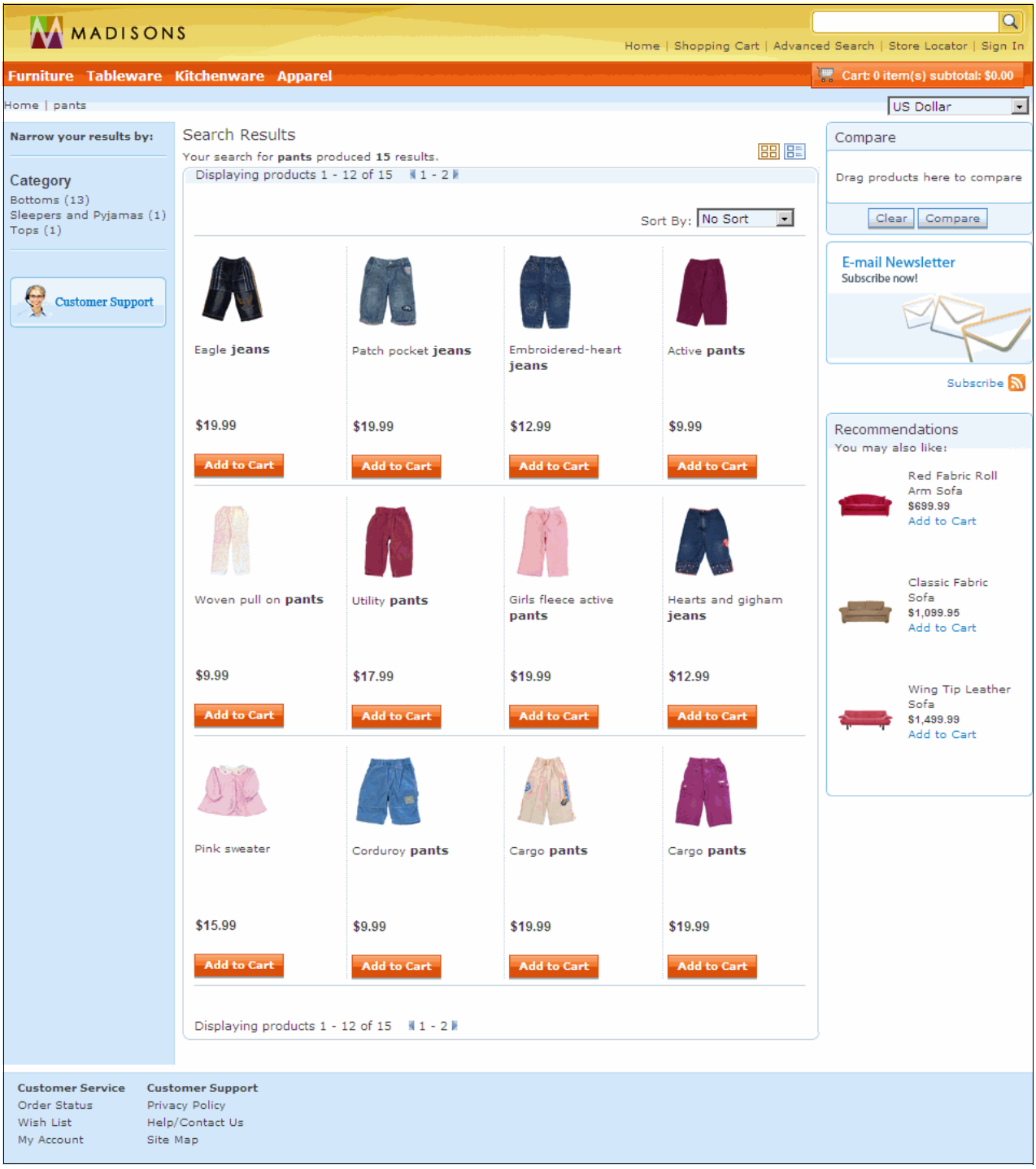


Figure 3-8 Search results for pants after defining jeans as a synonym

3.1.2 Replacement terms

Replacement terms are used to change the search results by replacing the search term before submitting the search term for the search. The search submission can include the defined changed search terms, while optionally discarding the submitted search term. The search results only include the search results for the replacement terms, helping to target certain product types over others.

Replacement terms are unidirectional: the rule applies only when searching for the search term, but not when searching for any of the replacement terms.

There are two types of replacement terms:

- *Also Search For*: Selecting this type of replacement term searches for both the input search terms and the replacement search terms.
- *Instead Search For*: Selecting this type of replacement term ignores the input search terms and instead searches for the replacement search terms.

In the current scenario, the business requires that we search for all products having the keyword `cup` instead of `mug` and the keyword `sofa` instead of `couch`. Before defining replacement terms, we show the search results in the storefront for the search term `mug` in Figure 3-9 on page 19 and the search results in the storefront for the search term `couch` in Figure 3-10 on page 20.

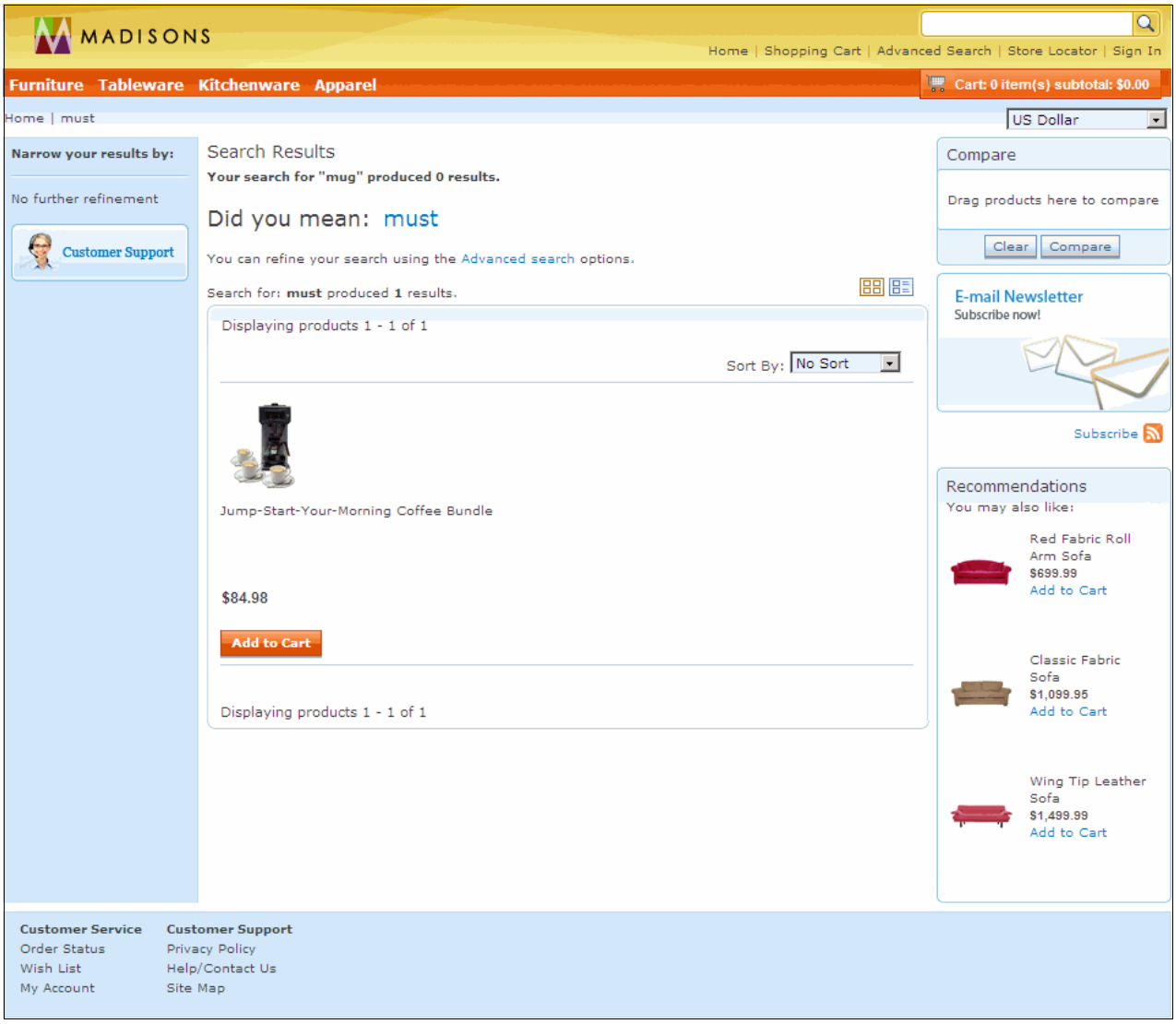


Figure 3-9 Search results for the search term mug before adding the replacement term

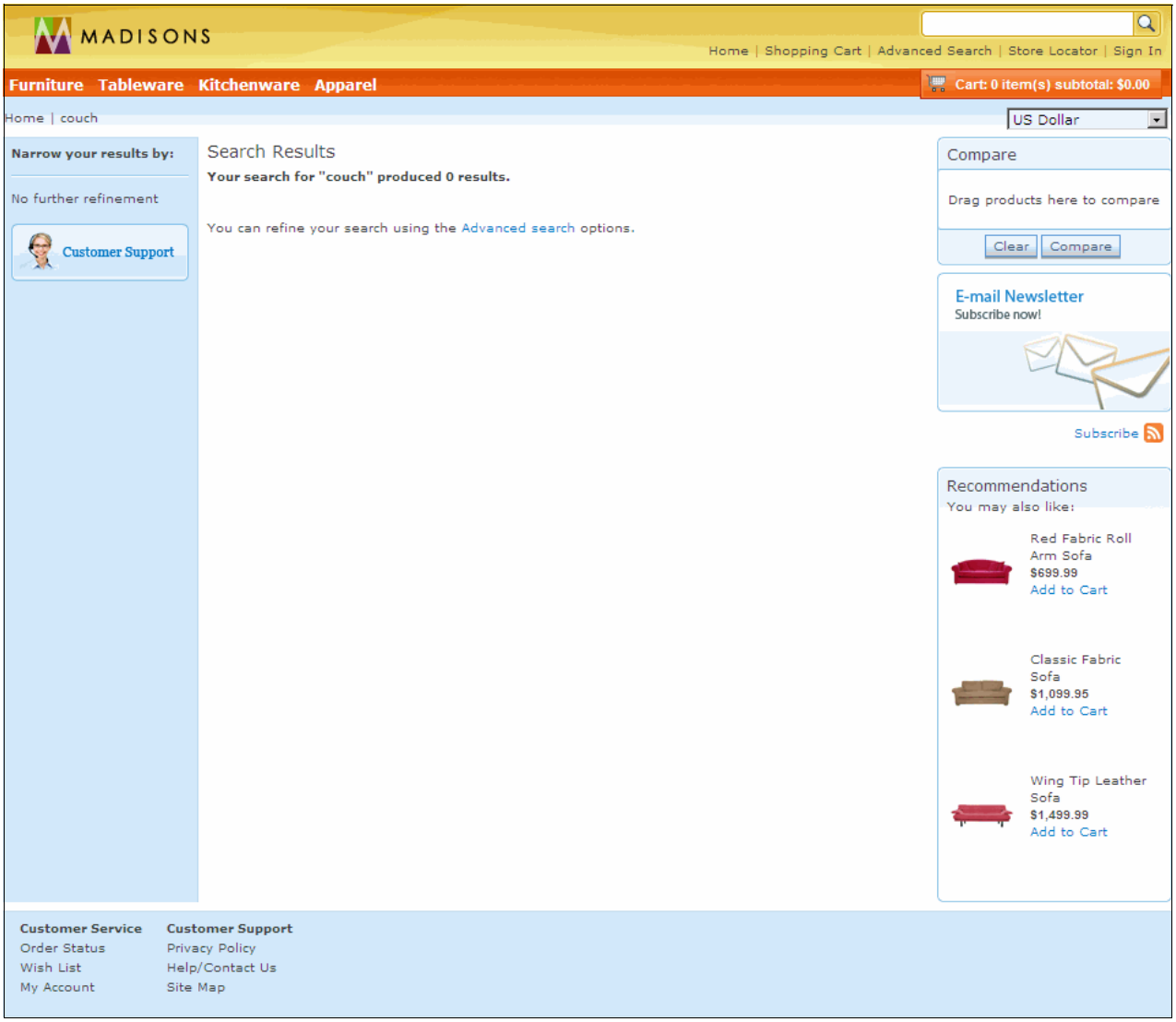


Figure 3-10 Search results for the search term couch before adding the replacement term

Follow these steps to implement the replacement term function:

1. Complete the steps in “Open search term associations” on page 12.
2. Click the **Replacements** tab.
3. Click **Create New Replacement**, as shown in Figure 3-11 on page 21.

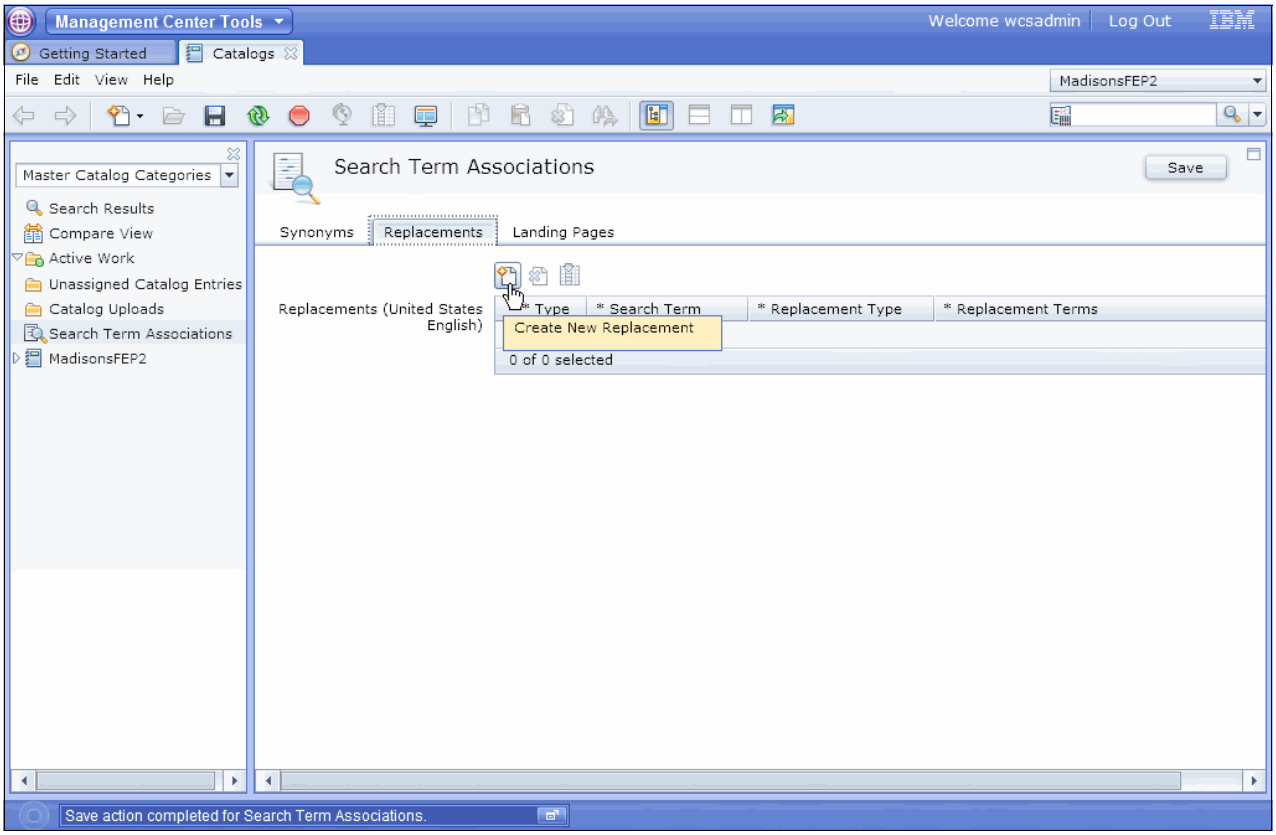


Figure 3-11 Create a new replacement term

4. Enter the following values, as shown in Figure 3-12:

   a. For Search Term, enter `mug`.
   b. For Replacement Type, select **Instead search for** from the drop-down list box.
   c. For Replacement Terms, enter `cup`.

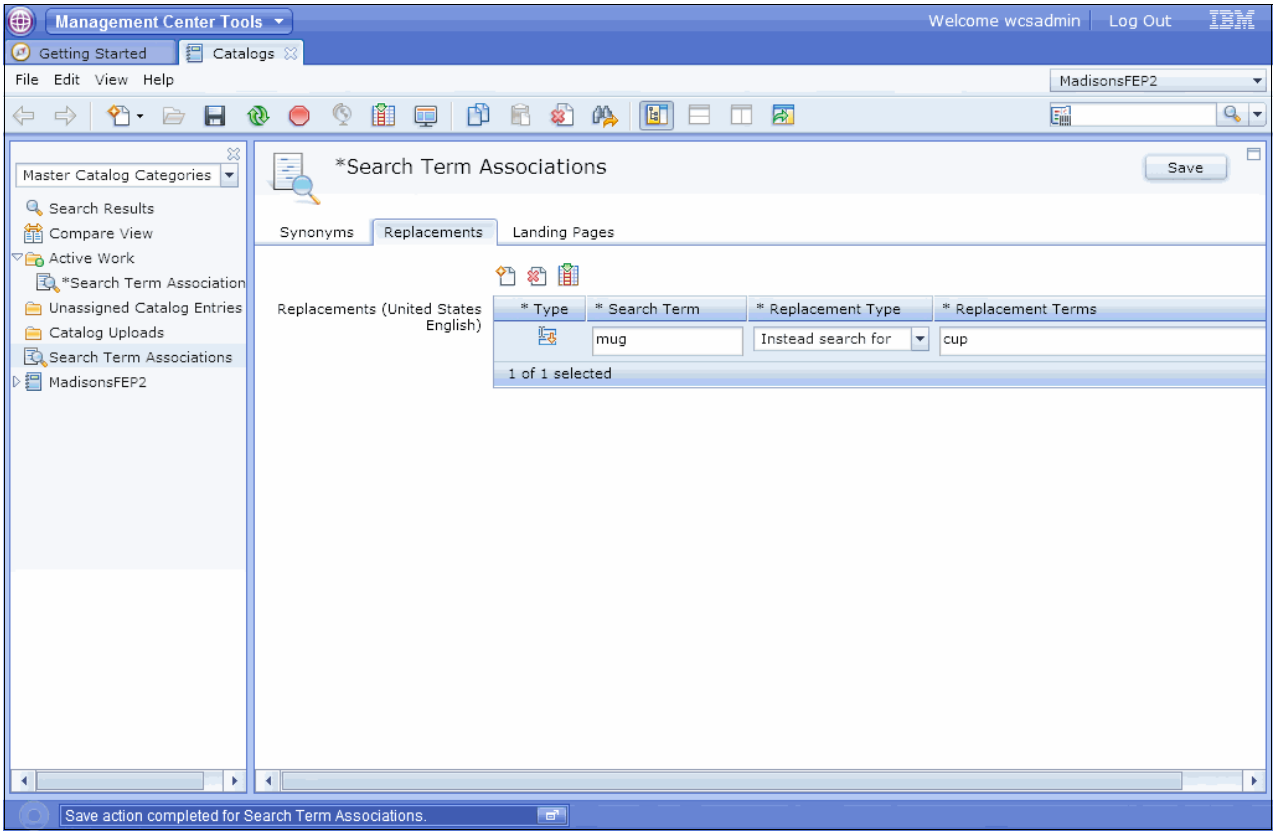


Figure 3-12 Entering replacement term cup for the term mug

5. Repeat Step 3 on page 20 and 4 on page 22 with the following values, as shown in Figure 3-13:
   a. For Search Term, enter `couch`.
   b. For Replacement Type, select **Also search for** from the drop-down list box.
   c. For Replacement Terms, enter `sofa`.

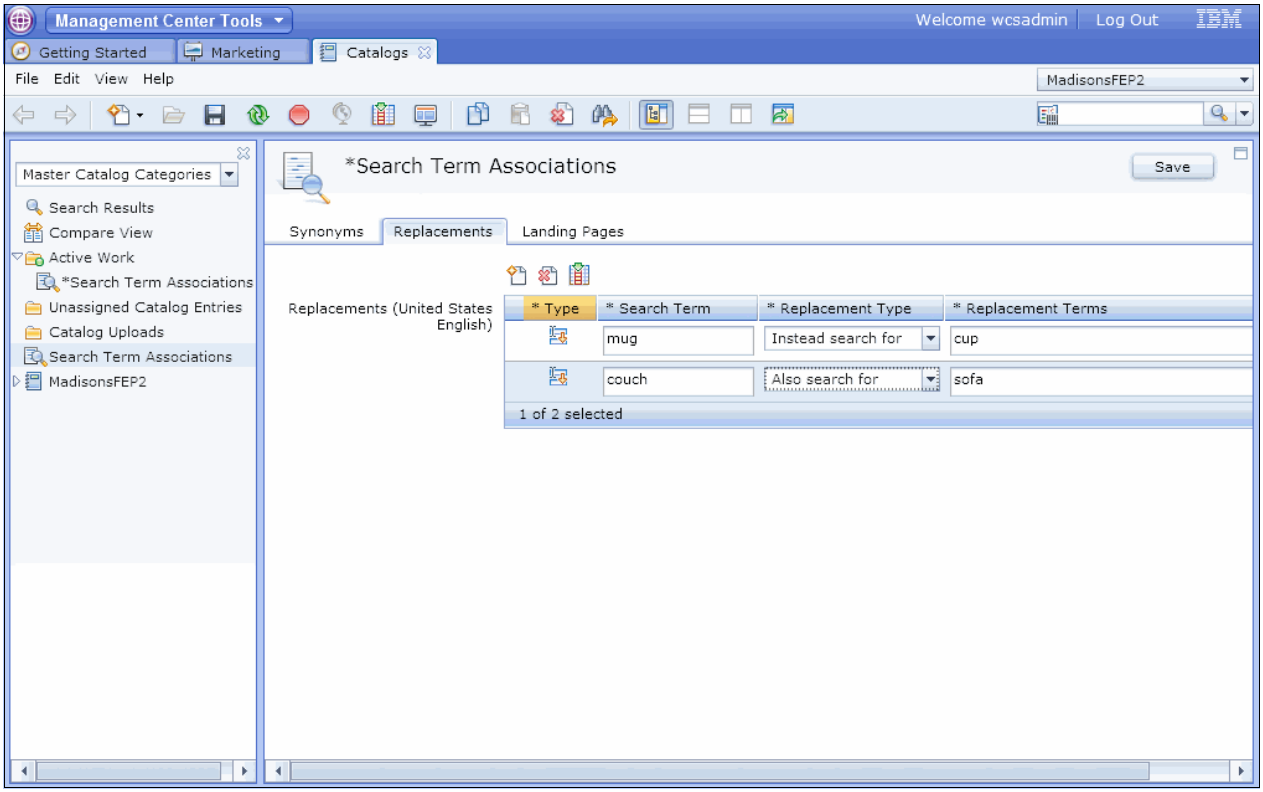


Figure 3-13 Entering the replacement term sofa for the term couch

6. Click **Save**.

In the storefront, if the customer now enters the search term, the customer gets the results for `cup`, as shown in Figure 3-14 on page 24. And if the customer searches for `couch`, the customer gets the results for `sofa`, as shown in Figure 3-15 on page 25.

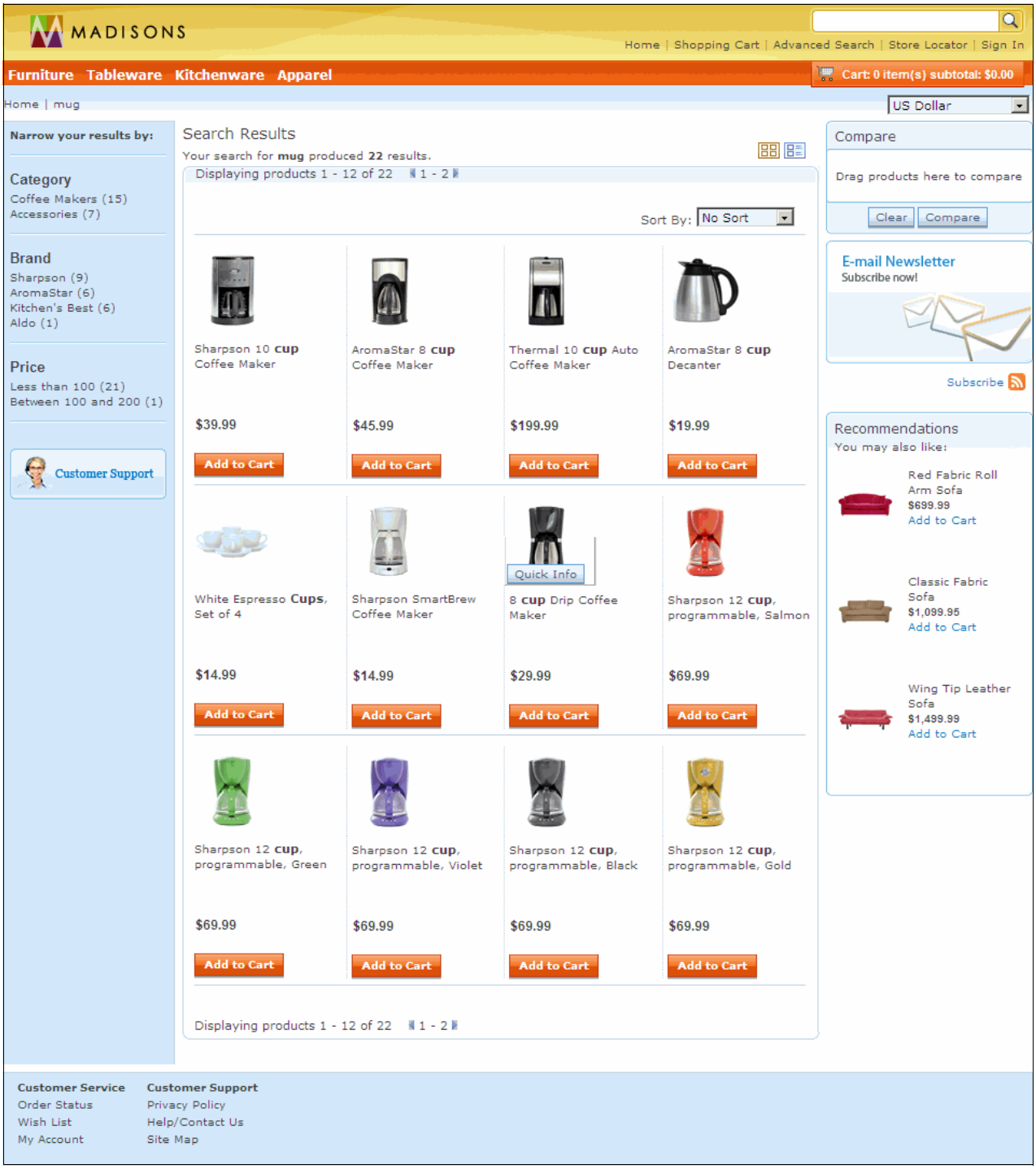


Figure 3-14 Search results for mug after adding the replacement term

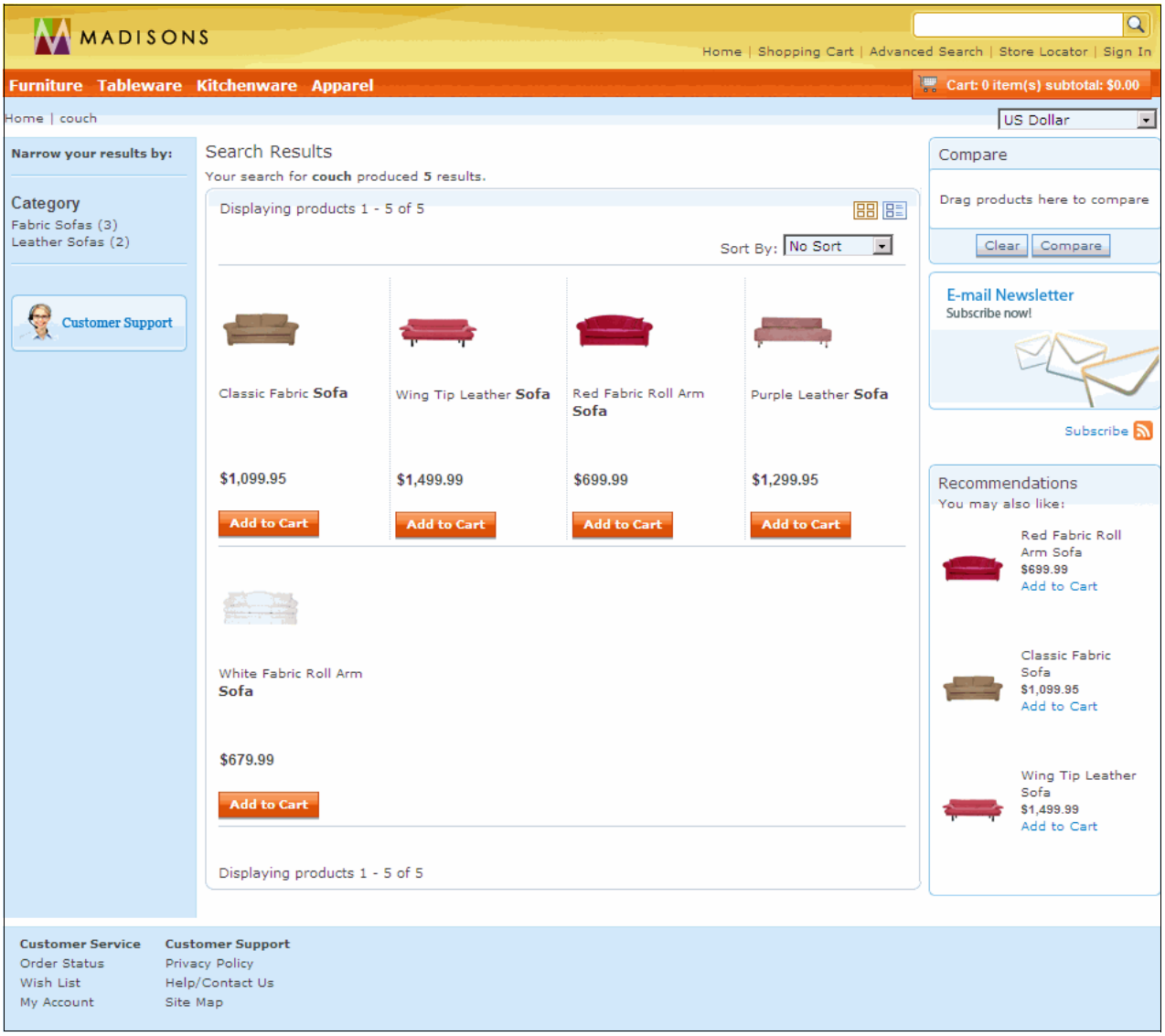


Figure 3-15 Search results for couch after adding the replacement term

3.1.3 Landing Page tab

If a shopper searches for a keyword, you can define the resulting landing page as a page of your choice, such as the related category in the store, as opposed to the search term's Search Results page. Associating a landing page as the result for a search term result promotes certain products or activities in the store by directing shoppers to specific store pages that are based on their search submissions.

Follow these steps to add a landing page for the search term `coffeemaker`:

1. Complete the steps in "Open search term associations" on page 12.
2. Click the **Landing Pages** tab.
3. Click the **New** icon to create a new landing page row:

4. Enter the following values, as shown in Figure 3-16:
   a. For Search Terms, enter `coffeemaker`.
   b. For Landing Page, enter `SearchLandingPage1`.

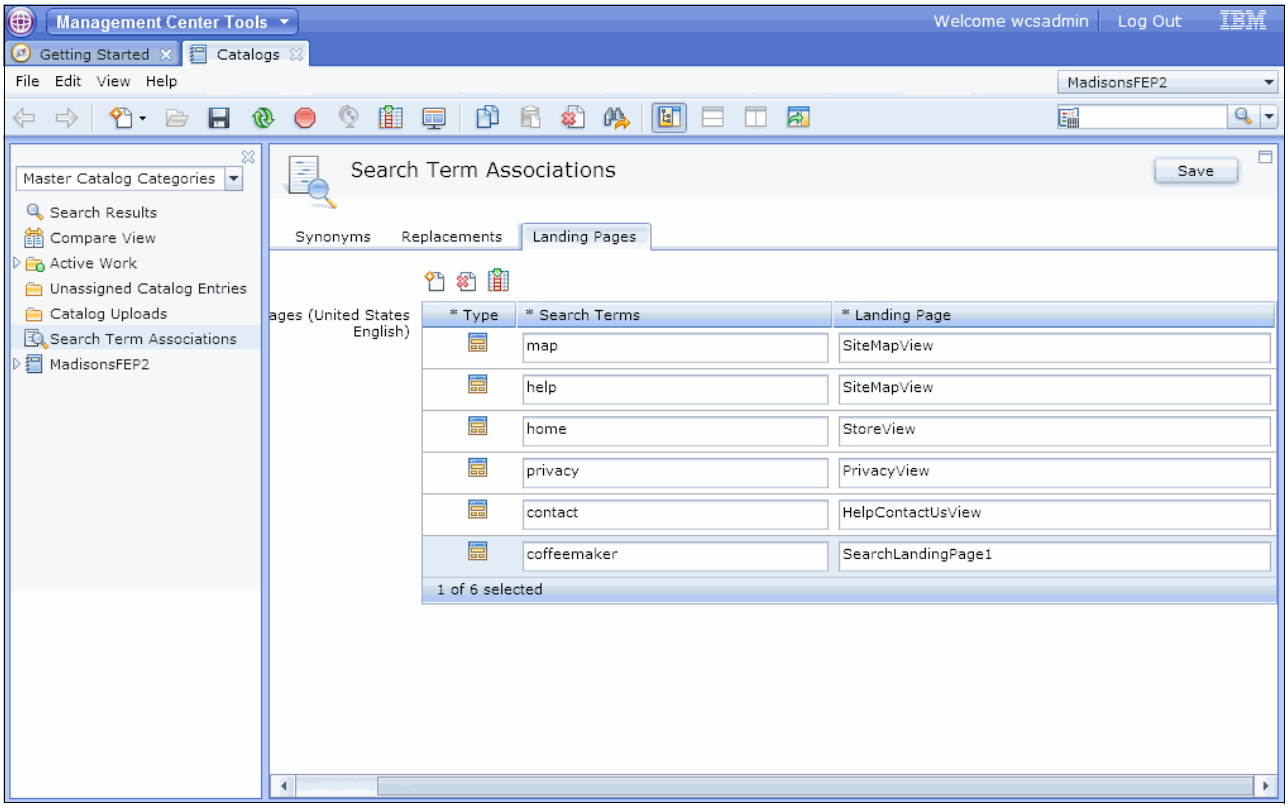


Figure 3-16 Configure the landing page

Important: You must create the landing page view and it must be available before you can configure it, as shown in Figure 3-16 on page 26.

When a shopper enters the keyword `coffeemaker` in the storefront, instead of the search results, the shopper is shown the Coffee Makers landing page, as shown in Figure 3-17 on page 27.

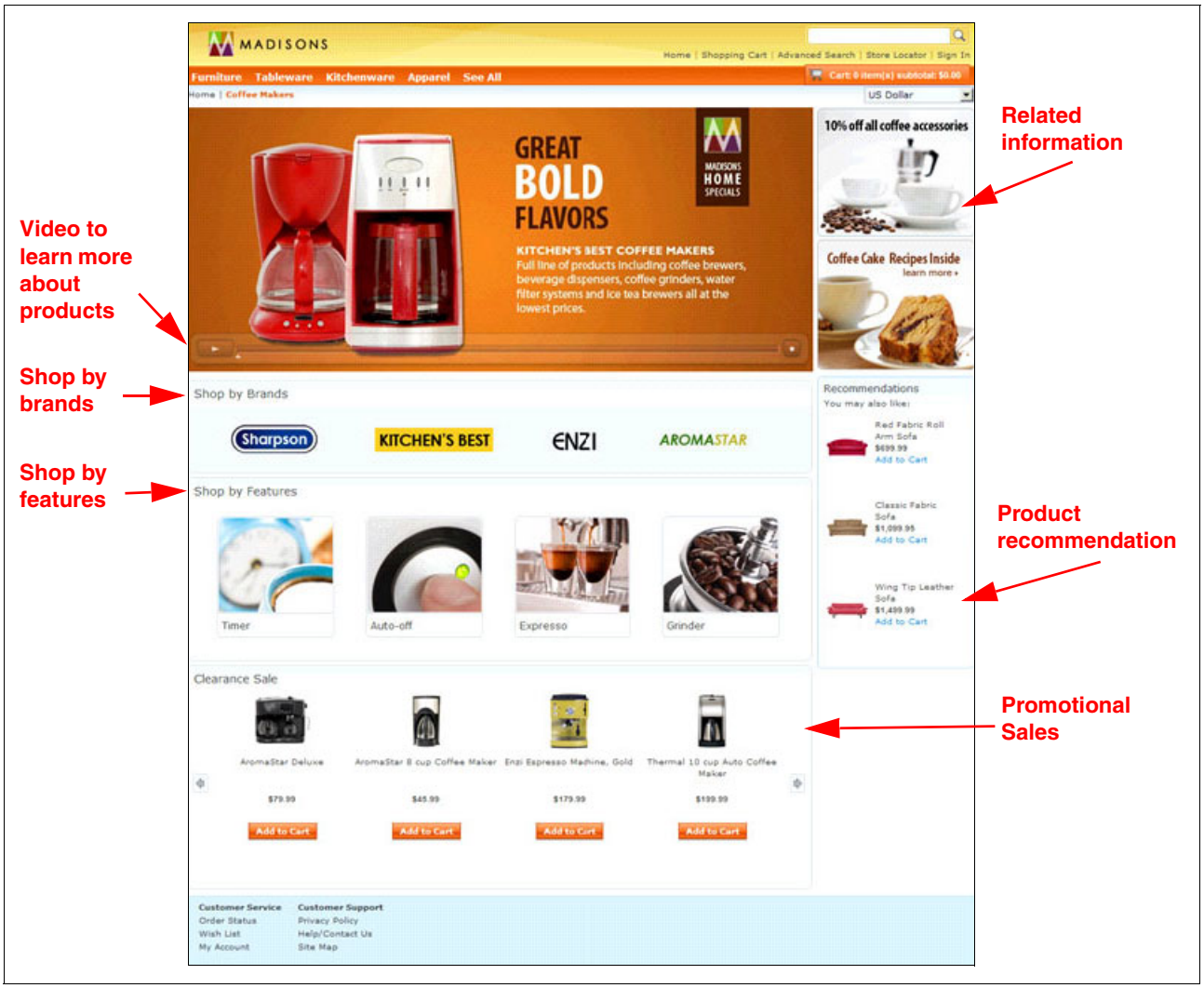


Figure 3-17 Search result for coffeemaker

3.2 Search-driven merchandising and marketing

You can write search rules by creating search rules, triggers, web activities, dialog activities, and so on. You can only write search rules if the following conditions are true:

- Feature Pack 2 is installed and the store enhancements feature is enabled.
- The Stores master catalog is indexed for WebSphere Commerce search.
- The search server and index structure are deployed and built.

Search Rule Builder

Search Rule Builder facilitates creating search rules and is displayed whenever the user clicks New Search Rule in the Marketing tool in the Management Center. You can only include one trigger for each search rule. Search rules must contain an action, which modifies the shopper's original search request to influence the order or content of the search result. Optionally, the rule can contain a target, which is placed to the left of an action. The target defines which customers experience the action.

As shown in Figure 3-18, the Search Rule Builder is split into three areas:

- Palette
- Work area
- General Properties view

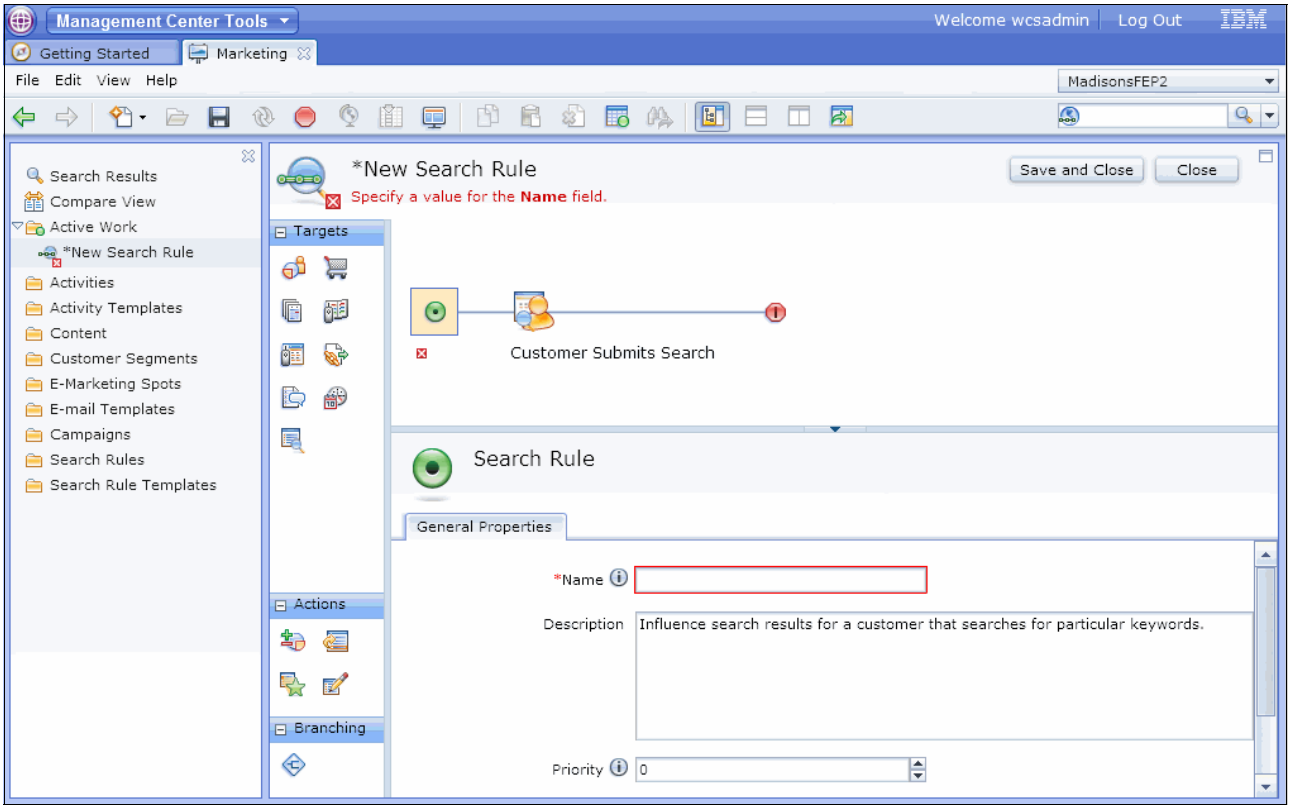


Figure 3-18 Search Rule Builder

Palette

The palette to the left of the work area contains targets, actions, and branching elements that you can drag into the work area:

- Targets: Targets define which customers experience the defined search rule. To include a target in a rule, drag the target from the palette into the work area flow in the Search Rule Builder. Place the target to the left of the action element that you want to target.

  The following targets are available:

  - Customer Segment targets customers who belong or do not belong to specific customer segments.
  - Shopping Cart targets customers whose current shopping cart contents meet certain criteria.
  - Purchase History targets customers whose purchase history in the store meets certain criteria.
  - Catalog Browsing Behavior targets customers who have browsed certain parts of the store catalog.
  - Online Behavior targets customers whose recorded activities on the store meet certain criteria.
  - External Site Referral targets customers who have entered the current session on the site from a link on an external site.

  - Social Commerce Participation  targets customers who have participated in social commerce on the site for a specified number of times.
  - Day/Time  specifies the days of the week or times of day that an activity is active.
  - Search Criteria and Result  targets customers who have selected specific search filters, or whose search results include specific catalog entries.

- Actions: Actions define what to do, based on the previous sequence of triggers and, optionally, targets in the search rule flow. To include an action in a search rule, drag the action from the palette into the work area of the Search Rule Builder. The following actions are available:
  - Add to or Remove from Customer Segment  adds the customer to or removes the customer from an existing customer segment that is specified.
  - Change Search Result Order  changes the position of certain results within the search results list. Catalog entries that meet certain criteria can be ranked higher or lower to promote specific catalog entries over others for a specific customer search.
  - Specify Top Search Results  elevates specific catalog entries to display at the top of the search results list.
  - Add or Replace Search Criteria  replaces search keywords that are submitted by the customer with alternative search keywords.
- Branches: A Branch element is an element that you can add to a Web or Dialog activity to split a single path into two or more paths. When a customer experiencing the activity reaches the Branch element, the customer proceeds down one or more of those paths based on the criteria that you define. As a result, a single activity can have multiple outcomes, and you can target each outcome to a separate set of customers.

Work area

The work area in the upper right displays the search rule as a horizontal line that represents the search rule flow.

The only trigger  for all search rules is “Customer Submits Search”, which is added by default in all search rules into the work area. This option triggers the rule to start or continue. You need to define the search term matching rule for this trigger. When setting up this trigger, you must specify which keywords the customer submitted for the search. The keyword matching rules allow you to match a search phrase that contains specific keywords. The following matching rules are available:

- Search keyword or phrase can be anything.
- Search keyword or phrase is exactly one of the following values.
- Search phrase starts with one of the following words.
- Search phrase contains one of the following words.
- Search phrase ends with one of the following words.

General Properties view

The General Properties view in the lower right displays the properties of the search rule or the selected target, action, or branching.

Search rule evaluation

Several search rules can run at the same time for separate keywords. The search rules are evaluated and run in the following sequence:

- Search rules that target specific keywords or phrases are evaluated and run in order according to their Priority value.
- Search rules that target all keywords are evaluated and run in order according to their Priority value.

Search rules: A search rule influences the search results only if the customer reaches the action element in the search rule flow. If the search rule flow contains target elements before the action element, the customer must meet the target criteria before the search rule can influence the search result.

You can create multiple search rules with separate actions to be executed for a single search request. Table 3-1 summarizes the compatibility characteristics of the search rule actions. The actions that are listed in the first column of the table are run as part of a higher priority rule than the actions in the first row and, therefore, are run first.

Table 3-1 Search rule evaluation

Action executed in a lower priority rule	Action executed in higher priority rule				
	Change Search Result Order (sorting)	**Change Search Result Order (ranking)**	**Specify Top Search Results**	**Add or Replace Search Criteria (add)**	**Add or Replace Search Criteria (replace)**
Change Search Result Order (sorting)	The result set is sorted according to the higher priority rule sorting criteria first, and lower priority rule sorting criteria second.	The search result sorting takes priority over ranking.	The search result sorting takes priority over the specified top search result.	The search result scoped by the Add or Replace Search Criteria action is sorted according to the sorting criteria specified in the Change Search Result Order action.	The search results for the replaced keyword are sorted.
Change Search Result Order (ranking)	The search result ranking takes priority over sorting.	The ranking criteria from both actions are used. In the case of a collision when both actions have the same ranking criterion, the action from the higher priority rule takes precedence.	Search results are ordered, but the top catalog entries are still displayed at the top of the search result.	The search result scoped by the Add or Replace Search Criteria action is ordered according to the ranking criteria that is specified in the Change Search Result Order action.	The search results for the replaced keyword are ordered according to the ranking criteria.

Action executed in a lower priority rule	Action executed in higher priority rule				
	Change Search Result Order (sorting)	**Change Search Result Order (ranking)**	**Specify Top Search Results**	**Add or Replace Search Criteria (add)**	**Add or Replace Search Criteria (replace)**
Specify Top Search Results	The top search result takes priority, and the search results are not sorted.	The search results are ranked, but the top catalog entries are still displayed at the top of the search result.	Either action's specified catalog entries are displayed first in the search results, in the order of their relevancy to the shopper's search terms, and in the order specified in the action (lower priority rule or higher priority rule).	Scoping the result set might remove the catalog entries from the Specify Top Search Result action.	The search keyword is replaced, but the top catalog entries are still displayed at the top of the search result.
Add or Replace Search Criteria (add)	The search result scoped by the Add or Replace Search Criteria action is sorted according to the sorting criteria that is specified in the Change Search Result Order action.	The search result scoped by the Add or Replace Search Criteria action is ordered according to the ranking criteria that is specified in the Change Search Result Order action.	The filter is applied, which might result in removing the catalog entries from the Specify Top Search Result action.	The search criteria from both actions are used. In the case of a collision when both actions have the same search criterion, the action from the higher priority rule takes precedence.	The search results for the replaced keyword are scoped.
Add or Replace Search Criteria (replace)	The search results for the replaced keywords are sorted.	The search results for the replaced keyword are ordered according to the ranking criteria.	The search keyword is replaced, but top catalog entries are still displayed at the top of the search results.	The search results for the replaced keyword are scoped.	The action from the higher priority rule takes precedence.

Customers can optionally choose to further sort their search results in the storefront after their initial search results are displayed. In this case, the customer's sorting order takes priority over the action's sorting orders.

3.2.1 Change Search Result Order

This action in a search rule is used to change the position of certain results within the search results list. Catalog entries that meet certain criteria can be ranked higher or lower to promote specific catalog entries over others for a specific customer search.

When using this action in search rules, you must select one of the following ordering options:

- Change how initial search results are ranked
- Change how initial search results are sorted

> **Sorting order:** The sorting order specified by the shopper always takes precedence over the way that this action changes the search results order.

When ranking initial search results, you must specify a boost factor. You specify a *boost factor* as a positive number, which elevates catalog entries in the search results by increasing their relevancy scores. Store search results are ordered by default from the most relevant matches to the least relevant matches. Therefore, assigning higher boost factors typically corresponds to higher relevancy scores for catalog entries. As a result, the catalog entries with higher boost factors appear higher in store search results.

You can assign boost factors for the following ranking criteria:

- Manufacturer name
- Part number
- Manufacturer part number
- Catalog entry type
- Name
- Short description
- Category

> **Boost factors:** Increasing boost factors increases the relevancy scores of catalog entries. It does not, however, guarantee elevating the catalog entries to the top of the search results. The boost factor that you specify in this action determines the position of the specified results relative to the other search results. However, if you specify a high boost factor value, for example, `999`, the catalog entry likely appears at the top of the search result, provided that it is already returned in the initial search results.

Scenario one

In this scenario, a male customer searches for `glasses` and the search displays only wine glasses. For all other customers, the search results are displayed sorted by price in descending order.

Perform the following instructions to implement this scenario:

1. Log on to Management Center:

   `http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.management-center.doc/tasks/ttflogon.htm`

2. Open the **Marketing** tool, as shown in Figure 3-19 on page 33.

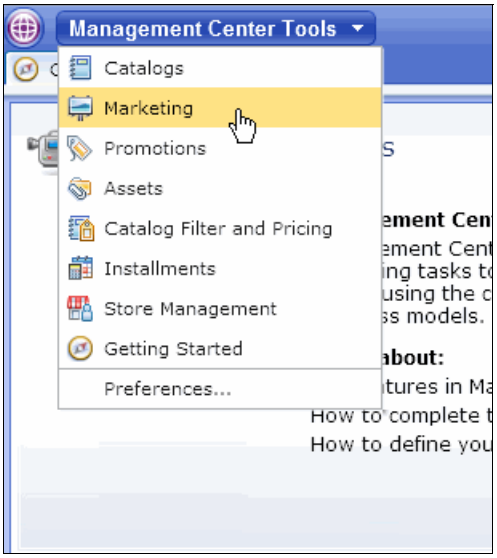


Figure 3-19 Open the Marketing tool

3. Select the store name **MadisonsFEP2** from the upper-right drop-down list box. Alternatively, you can set a store as the default store to be opened for all tools in the Management Center by adding it as a preference. Refer to the following link in the WebSphere Commerce Information Center:

   `http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.management-center.doc/tasks/ttfupdatepreference.htm`

   The rest of this book assumes that `MadisonsFEP2` has been set as the preferred store.

4. Select **Search Rule** from the Create New toolbar icon, as shown in Figure 3-20.

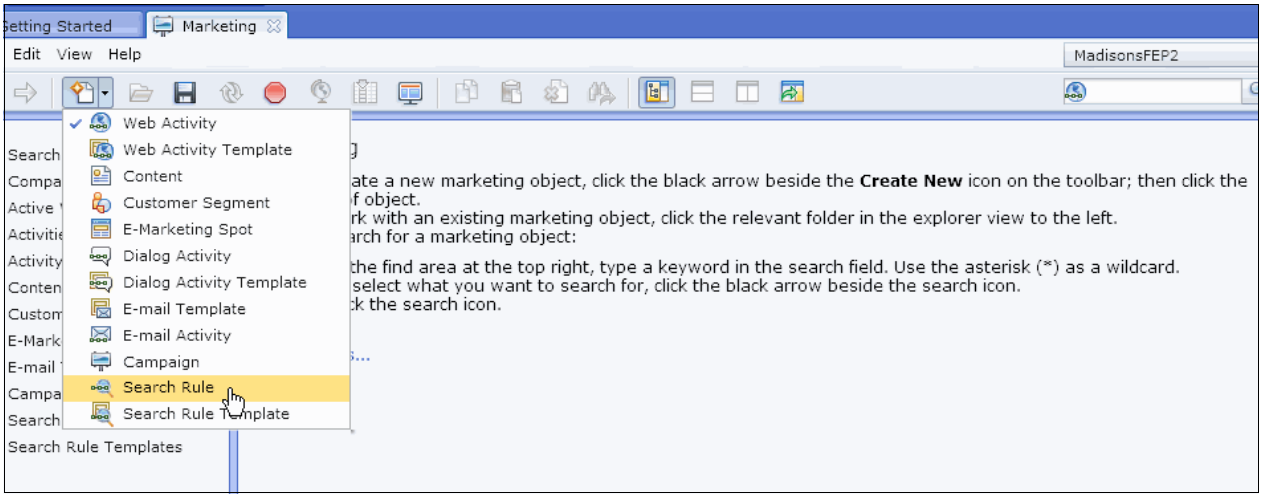


Figure 3-20 Select Search Rule

5. Select the standard template named **Change Search Result Order** in the pop-up window and click **OK**, as shown in Figure 3-21.

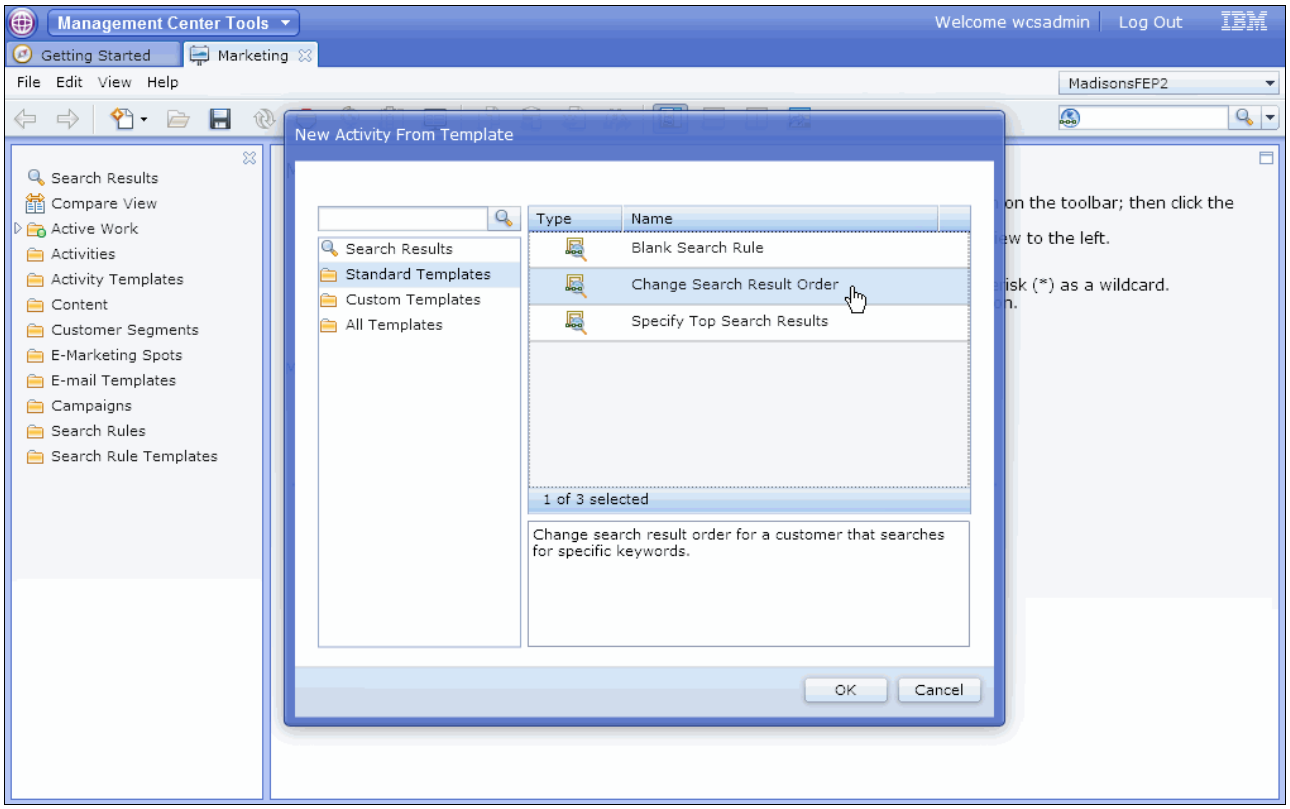


Figure 3-21 Click Change Search Result Order

6. Enter the Name of the search rule as `Glasses Search result order` on the General Properties tab, as shown in Figure 3-22 on page 35.

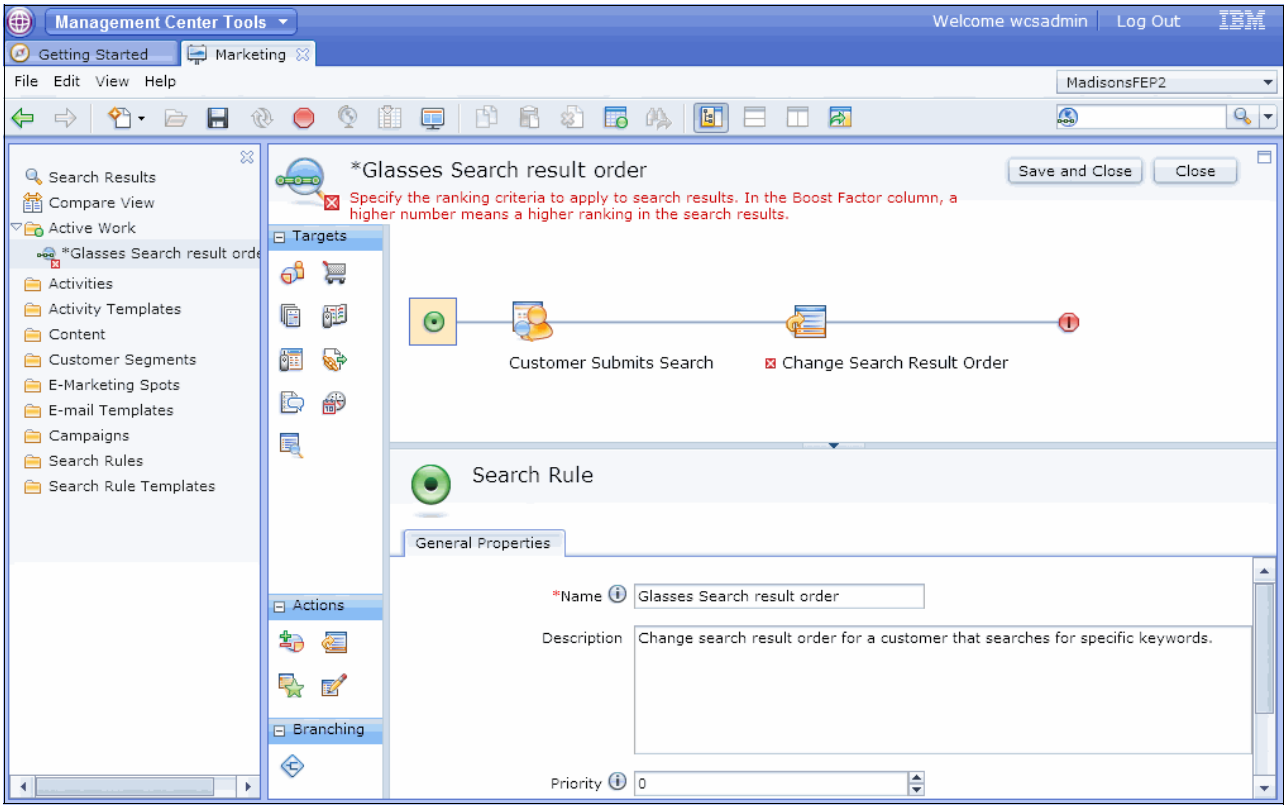


Figure 3-22 Enter the search rule name

7. Click the **Customer Submits Search** trigger in the work area.

8. Select the Matching rule as **Search keyword or phrase is exactly one of the following values**, as shown in Figure 3-23.

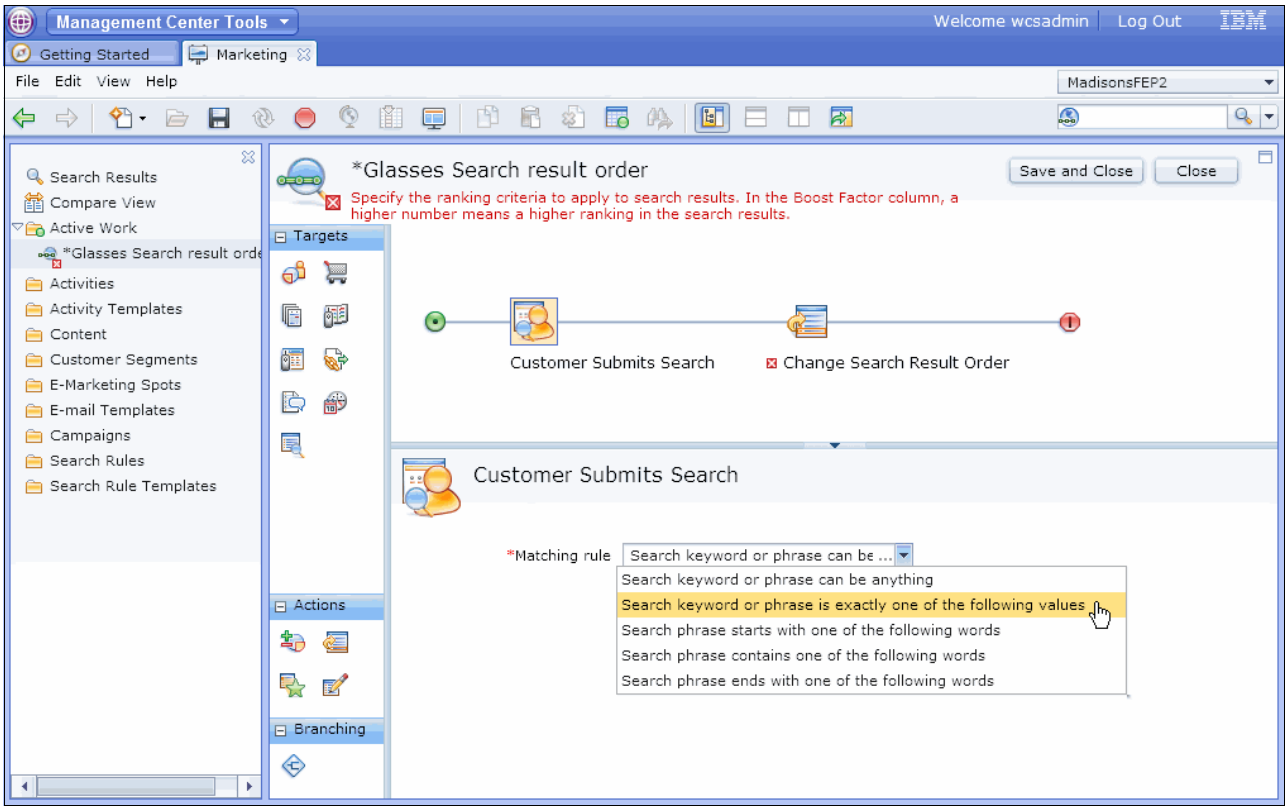


Figure 3-23 Select the Matching rule

9. Click the **Create New Keyword** icon and add `glasses` as a keyword, as shown in Figure 3-24.

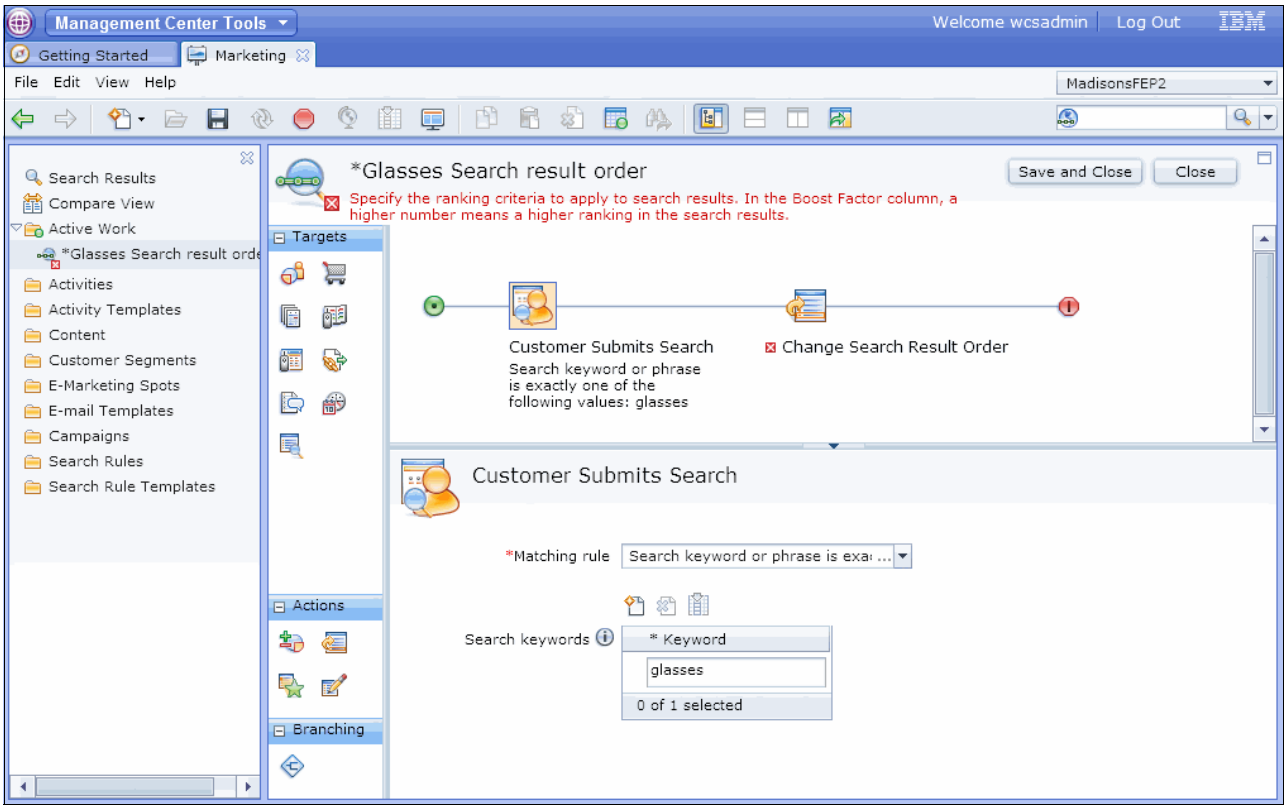


Figure 3-24 Add the keyword

10. Drag and drop the **Branching** icon and enter the following details, as shown in Figure 3-25:
    a. For Branch type, select **First path for which the customer qualifies**.
    b. For Paths Name, enter `Male Customers` and `All other customers`.

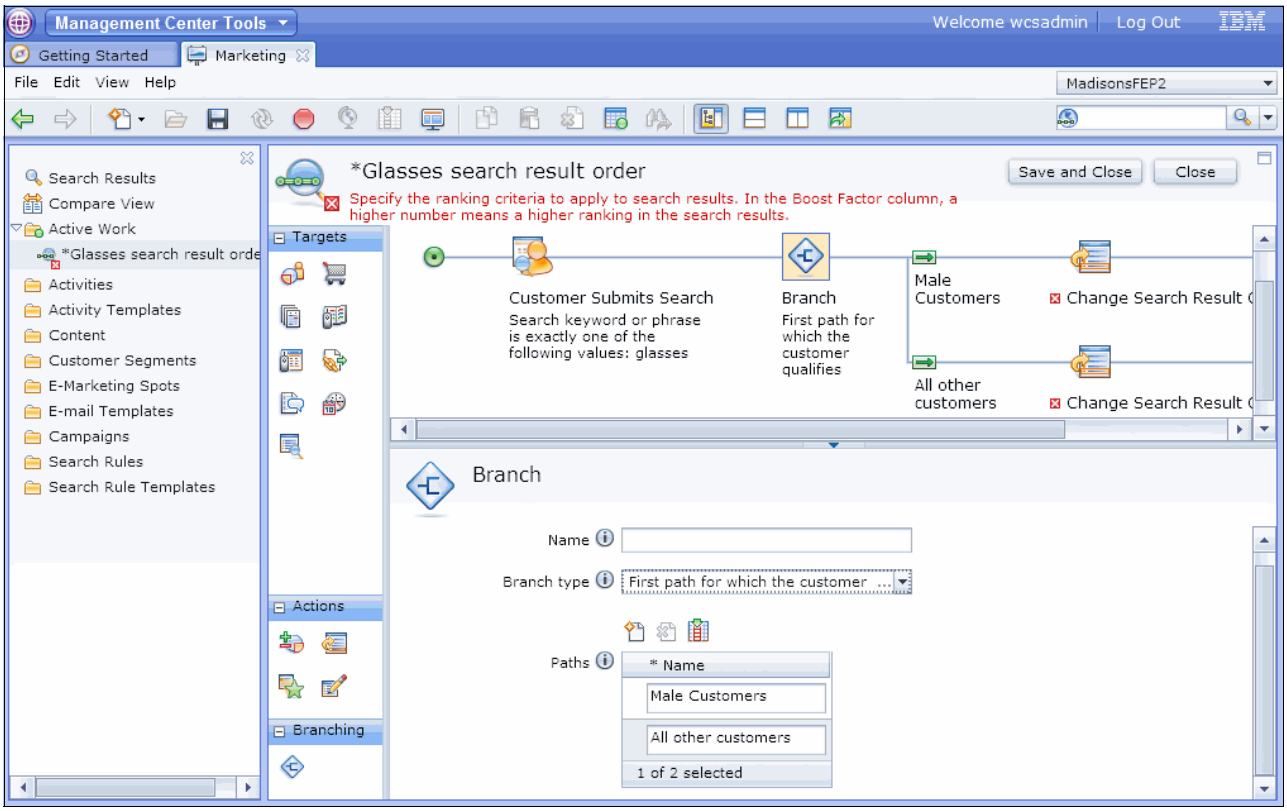


Figure 3-25 Add a Branch

11. Perform the following steps in Path 1 (Male Customers):

    a. Drag and drop the target **Customer Segment** icon and enter the following properties as shown in Figure 3-26:

        i. For Target customers, select **Who are in any of the following customer segments**.

        ii. In the search field for Customer segments, for Name, enter `Male Customers`.

        iii. Click **Find and Add**.

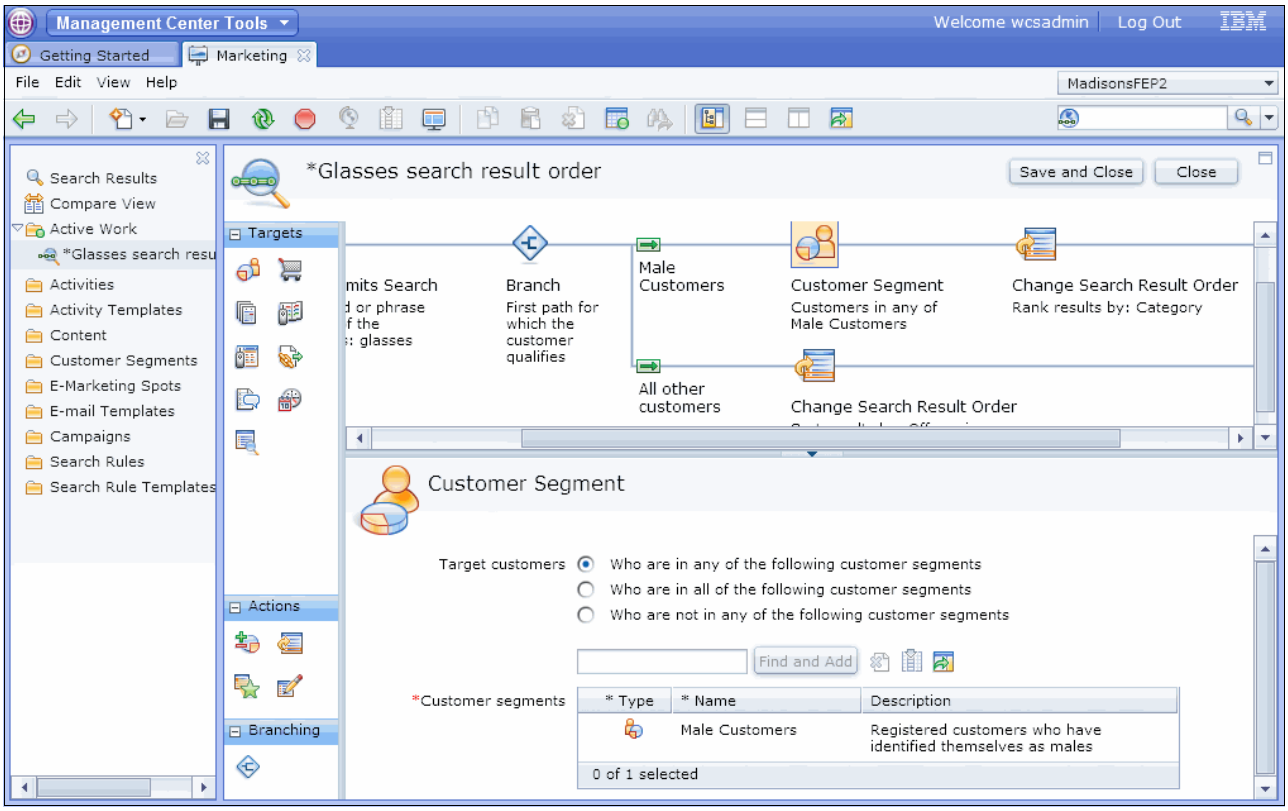


Figure 3-26 Add target Customer Segment

b. Under Actions, click the **Change Search Result Order** icon:

   i. In the Properties view, under Change Search Result Order, select **Change how initial search results are ranked**.

   ii. For the Ranking criteria, click the **New** icon and select **Category**, as shown in Figure 3-27.

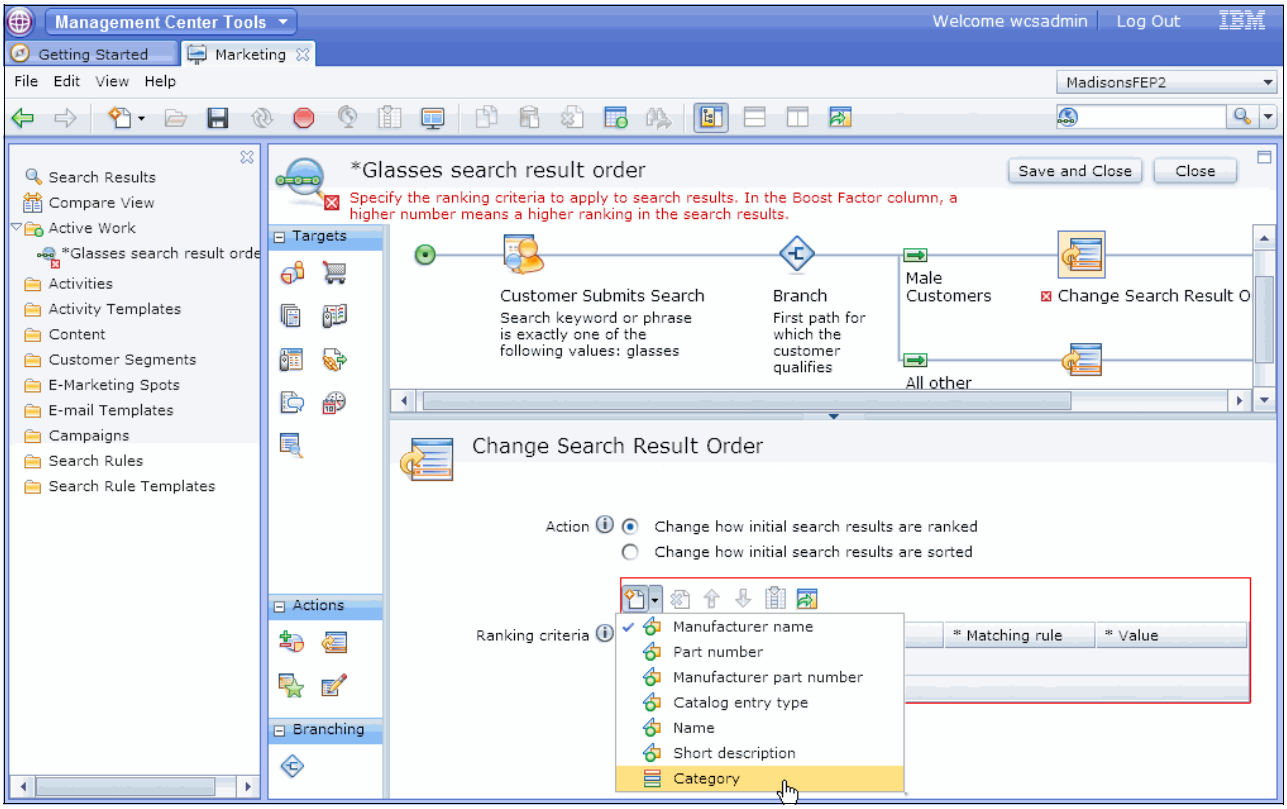


Figure 3-27 Add Ranking criteria

iii. In the pop-up Find and Add window, search the keyword `wine glasses` and click **OK**.

iv. Enter the boost factor `1.5`.

v. Select the matching rule **Matches**, as shown in Figure 3-28.

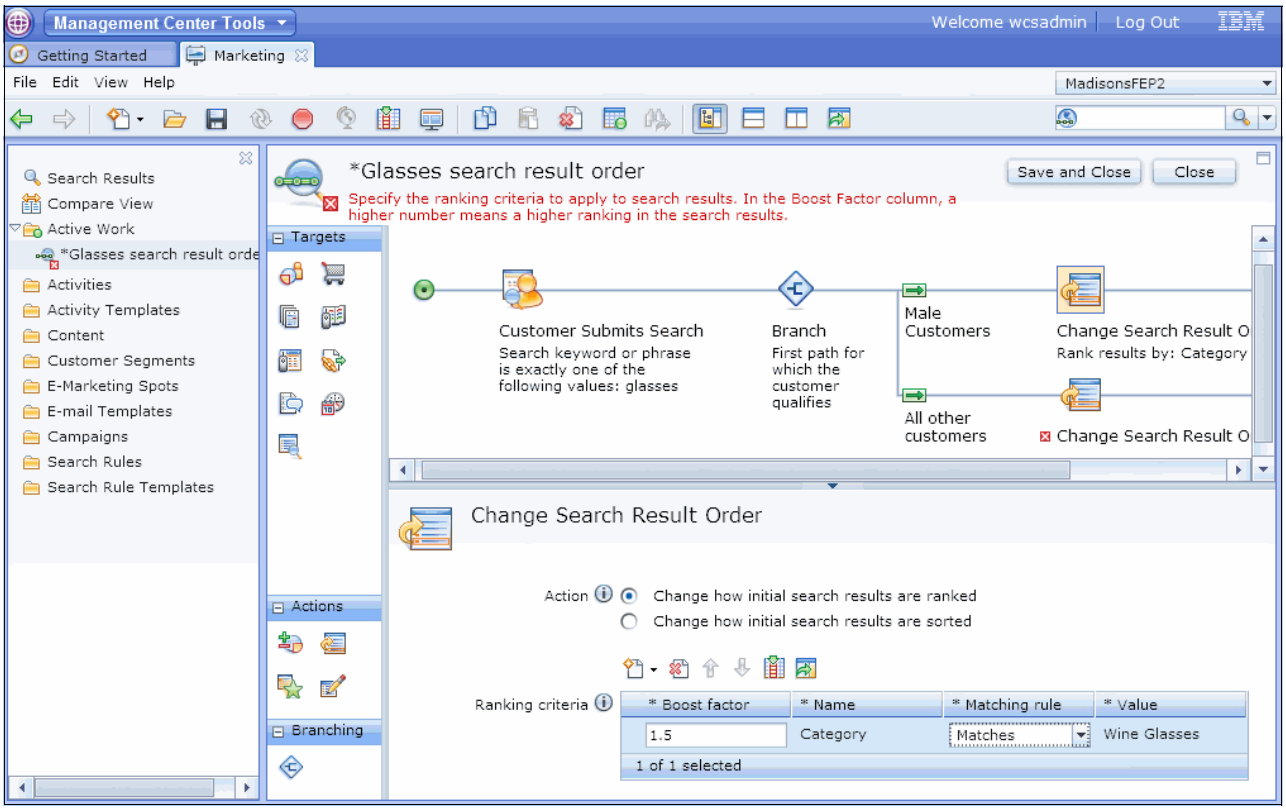


Figure 3-28 Select Matching rule

12. Perform the following steps in Path 2 (All other Customers), as shown in Figure 3-29:
    a. Click the **Change Search Result Order** action.
    b. In the Properties view, under Change Search Result Order, select **Change how initial search results are sorted**.
    c. In Sort criteria, click the **Create New** Sort criteria icon.
    d. Select **Offer price** in the Sort by column.
    e. Select **Descending** in the Sequence column.

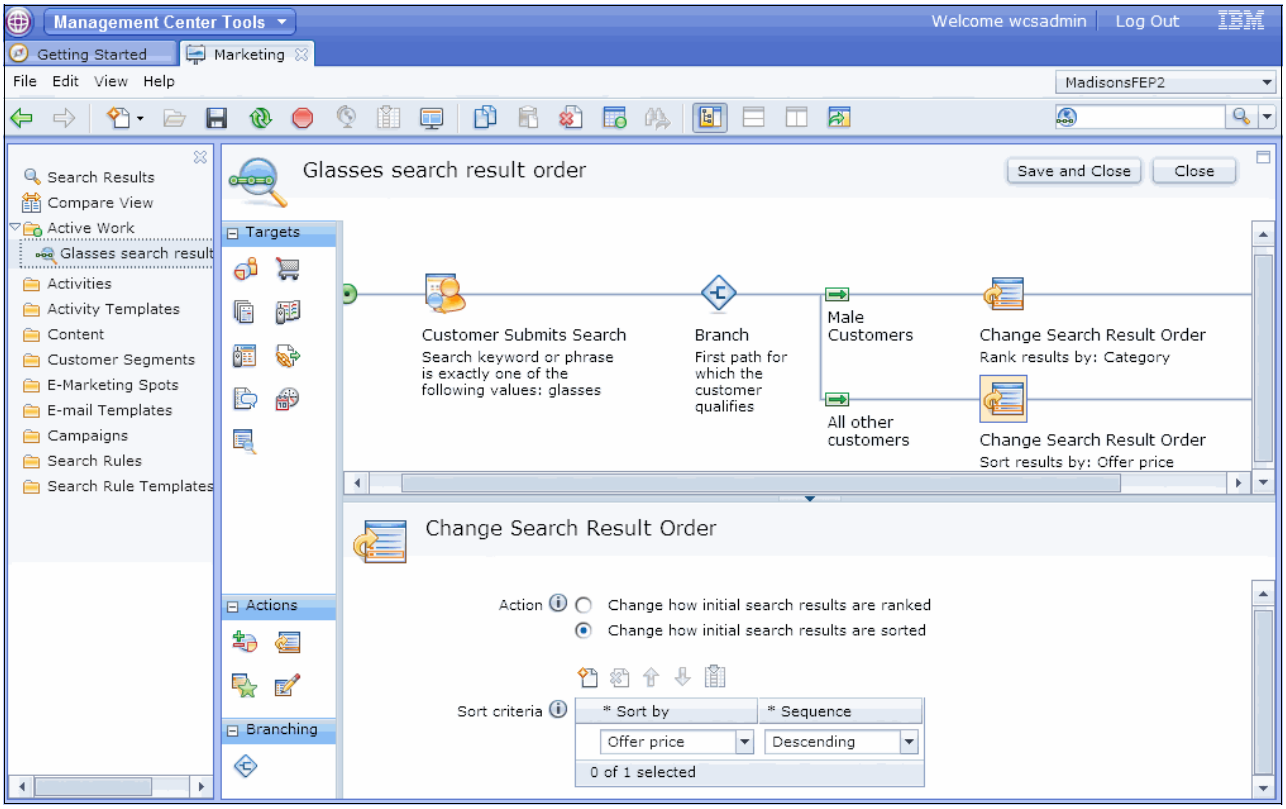


Figure 3-29 Update the properties for path 2

13. Click **Save and Close**.

14. The list of search rules is displayed. Right-click the current search rule and click **Activate**, as shown in Figure 3-30.

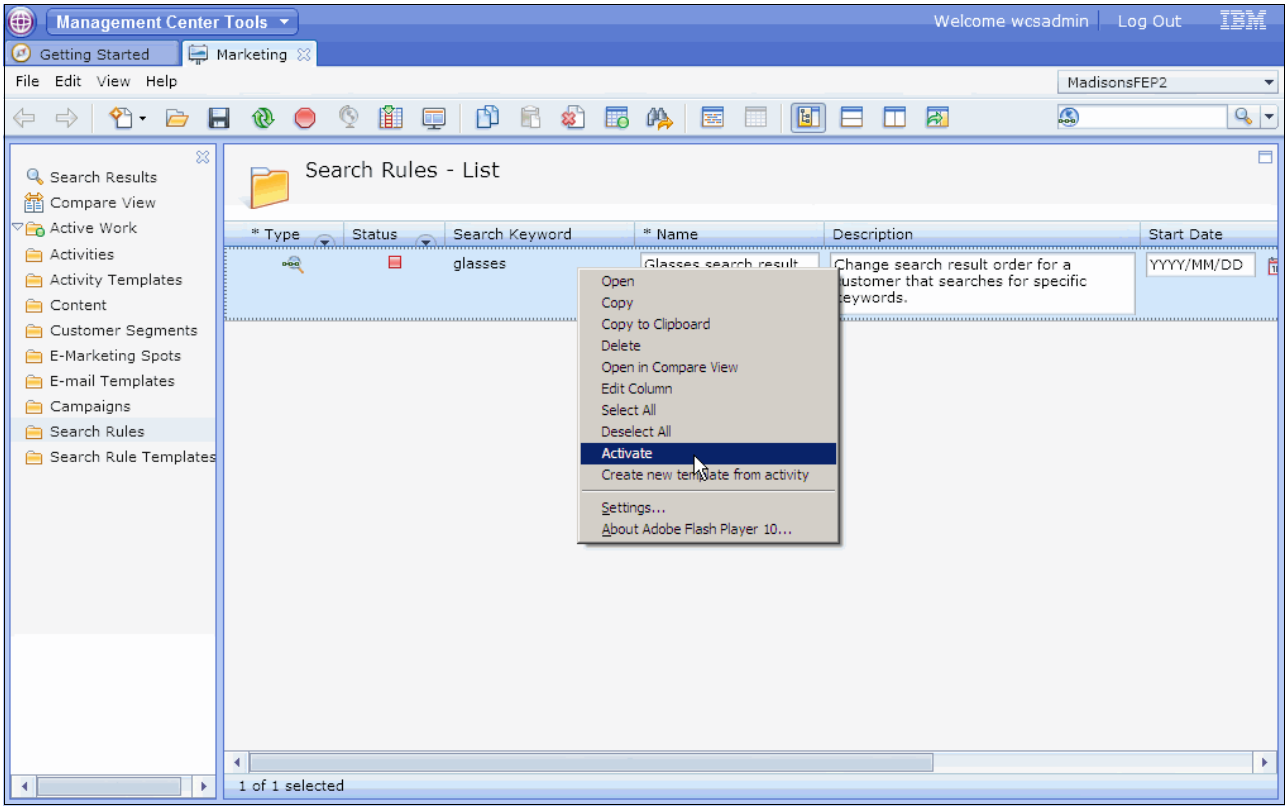


Figure 3-30 Activate the search rule

15. Search for `glasses` in the store as a guest user without logging in. The search results display the results sorted by price in descending order, as shown in Figure 3-31.

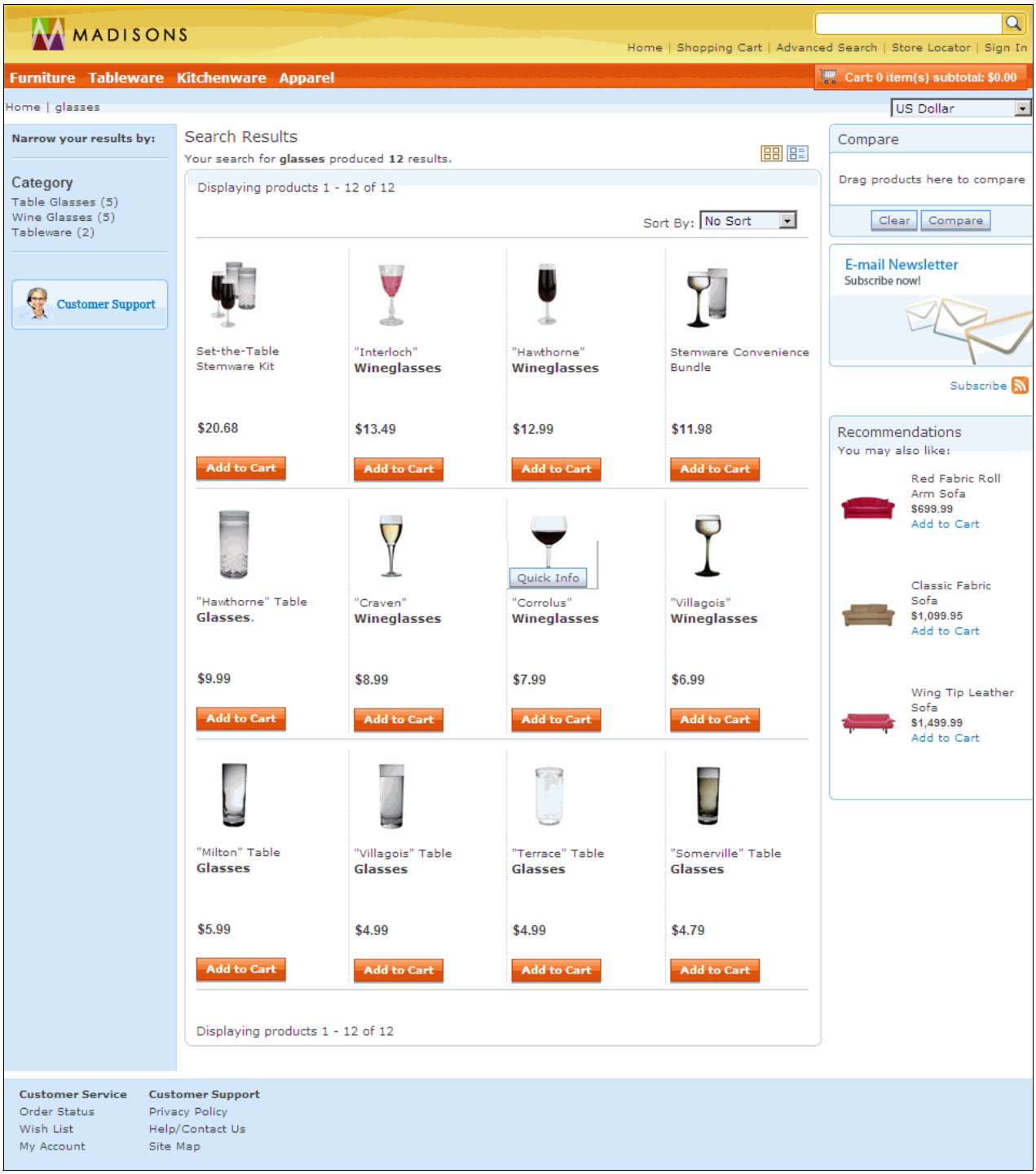


Figure 3-31 Search rule-based search results for all users

16. Log in to the store as a male customer and search for `glasses`. The search results rank wine glasses higher than other glasses, as shown in Figure 3-32.

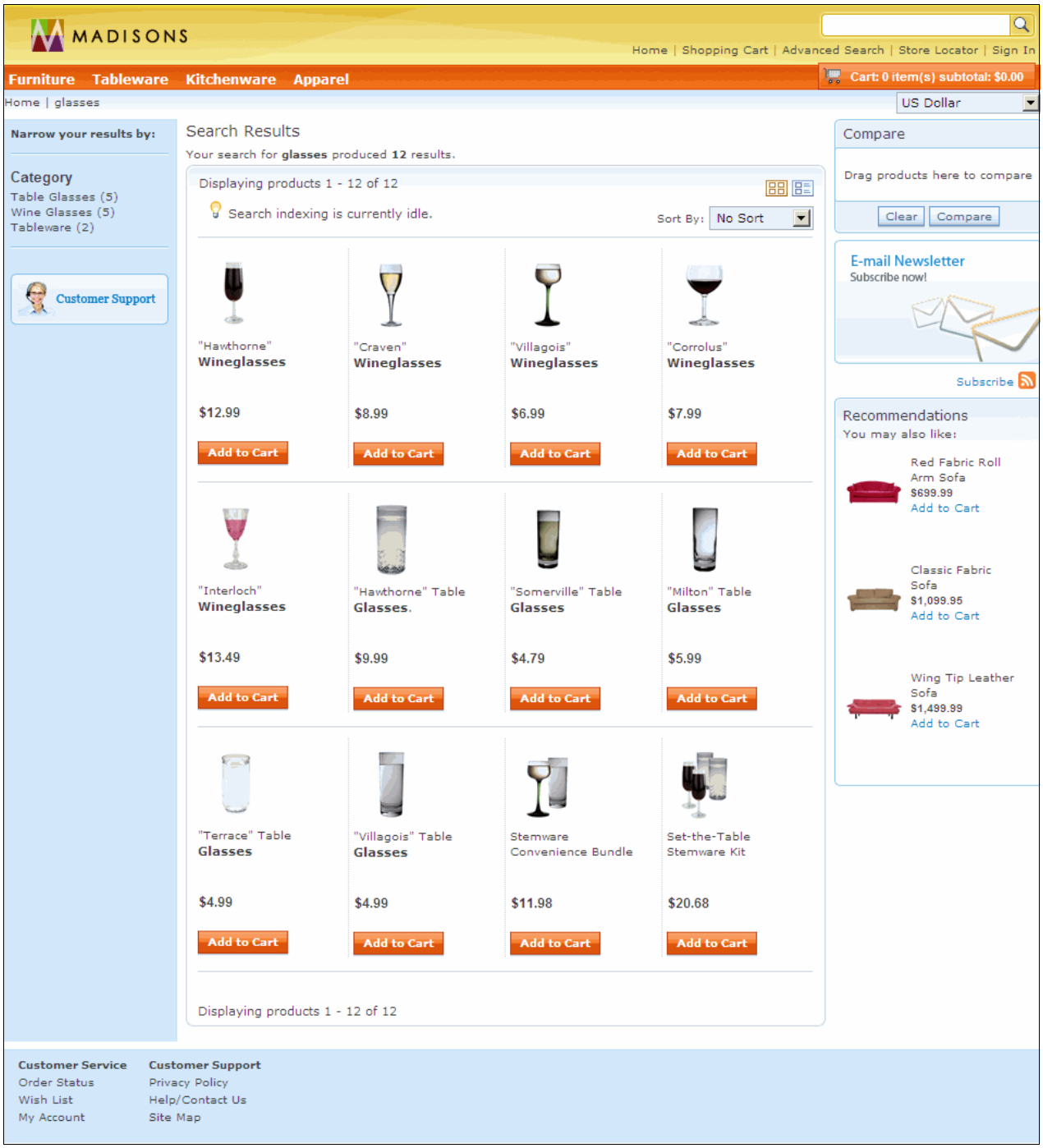


Figure 3-32 Search rule-based search results for male customers

Scenario two

For customers who viewed the coffee makers category at least three times within the last two days, boost the products from the coffee makers category when they search for `steel`.

Figure 3-33 on page 46 shows the search results for `steel` before boosting coffee makers.

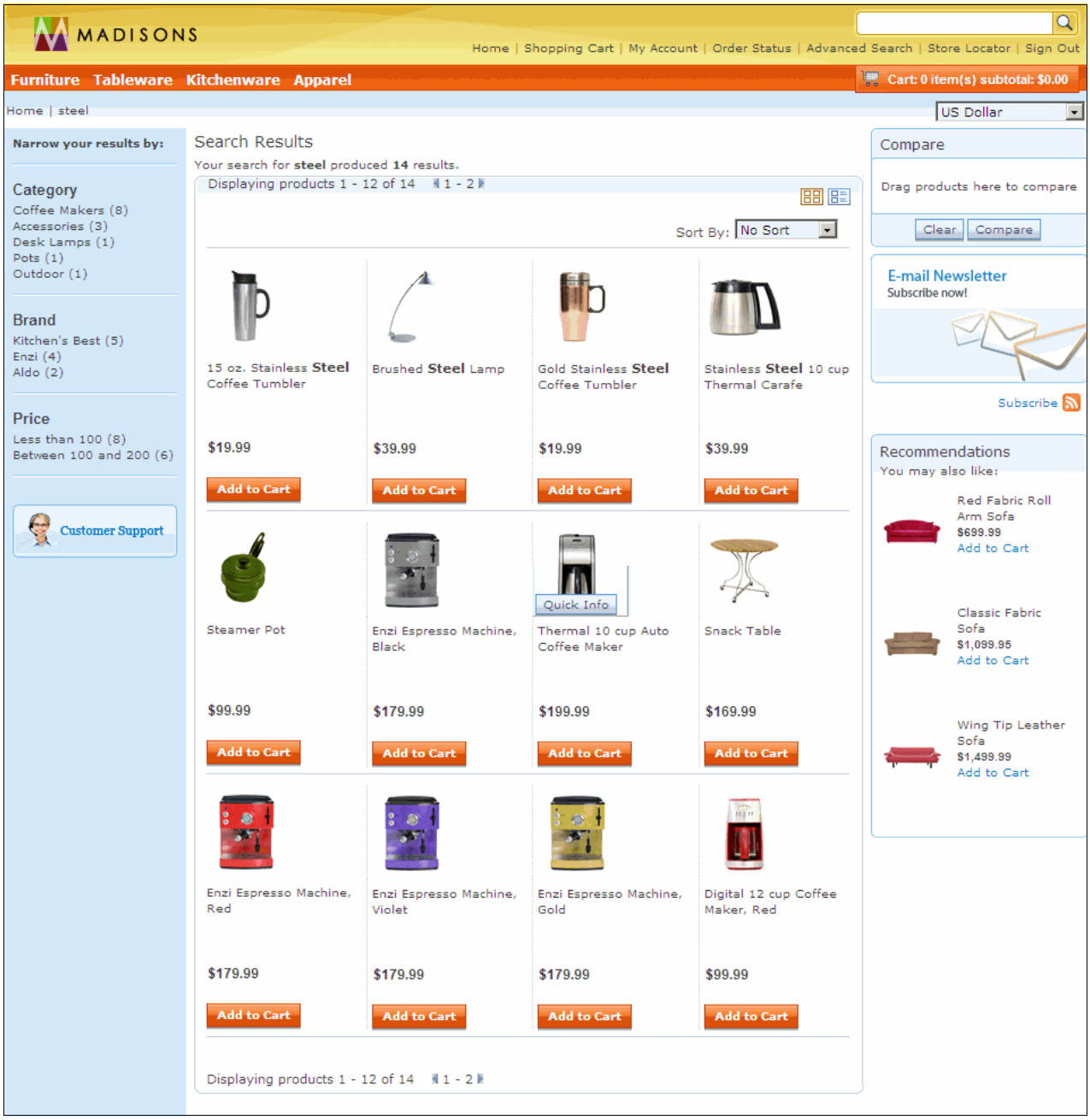


Figure 3-33 Search results for steel before creating the search rule

Follow these steps to boost coffee makers to address the requirement in the scenario:

1. Log on to the Management Center and open the **Marketing** tool.
2. Select **Search Rule** from the Create New toolbar icon.
3. Select the **Change Search Result Order** template in the pop-up window and click **OK**.

4. Enter the search rule properties, as shown in Figure 3-34.

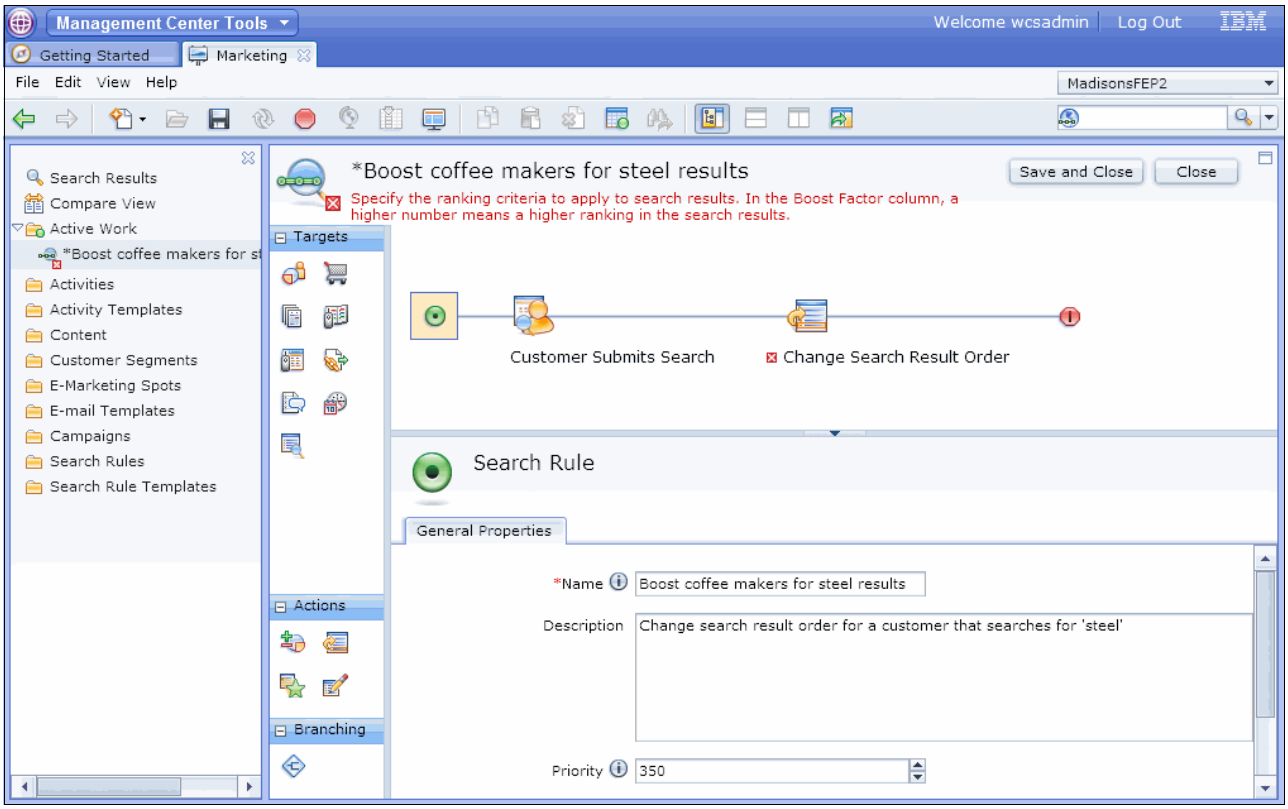


Figure 3-34 General Properties for the search rule

5. Click the **Customer Submits Search** trigger and enter the following values, as shown in Figure 3-35:
   a. For the Matching rule, select **Search keyword or phrase is exactly one of the following values**.
   b. For the Search keywords, enter `steel`.

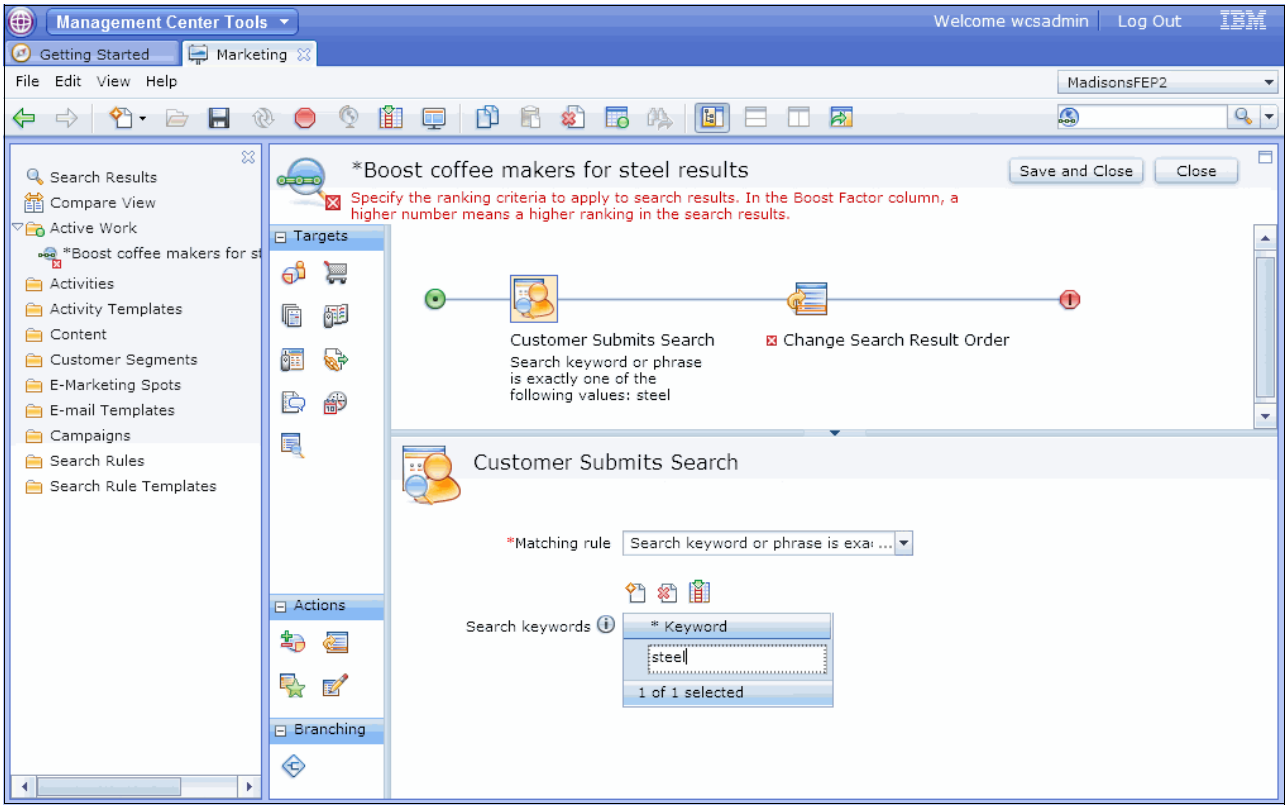


Figure 3-35 Properties for the trigger

6. Drag and drop the target **Catalog Browsing Behavior** icon into the work area after the trigger.

7. Click **Catalog Browsing Behavior** and enter the following values in the properties view, as shown in Figure 3-36:
   a. For Customer behavior, select **Customer viewed a category**.
   b. For Target customers, select **Who viewed any of the following categories and satisfy the following conditions**.
   c. For Categories, click **Find and Add**, enter coffee makers.
   d. Make sure that Include subcategories is checked.
   e. For Frequency, select **At least the following number of times**.
   f. For Times, select **3**.
   g. For Time frame, select **Within the following number of days**.
   h. For Days, select **2**.

Figure 3-36 Properties for the search rule target

8. Click the **Change Search Result Order** icon and enter the following values in the properties view, as shown in Figure 3-39 on page 51:
   a. For Action, select **Change how initial search results are ranked**.
   b. Complete these steps for the Ranking criteria:
      i. Click **Category**, as shown in Figure 3-37.

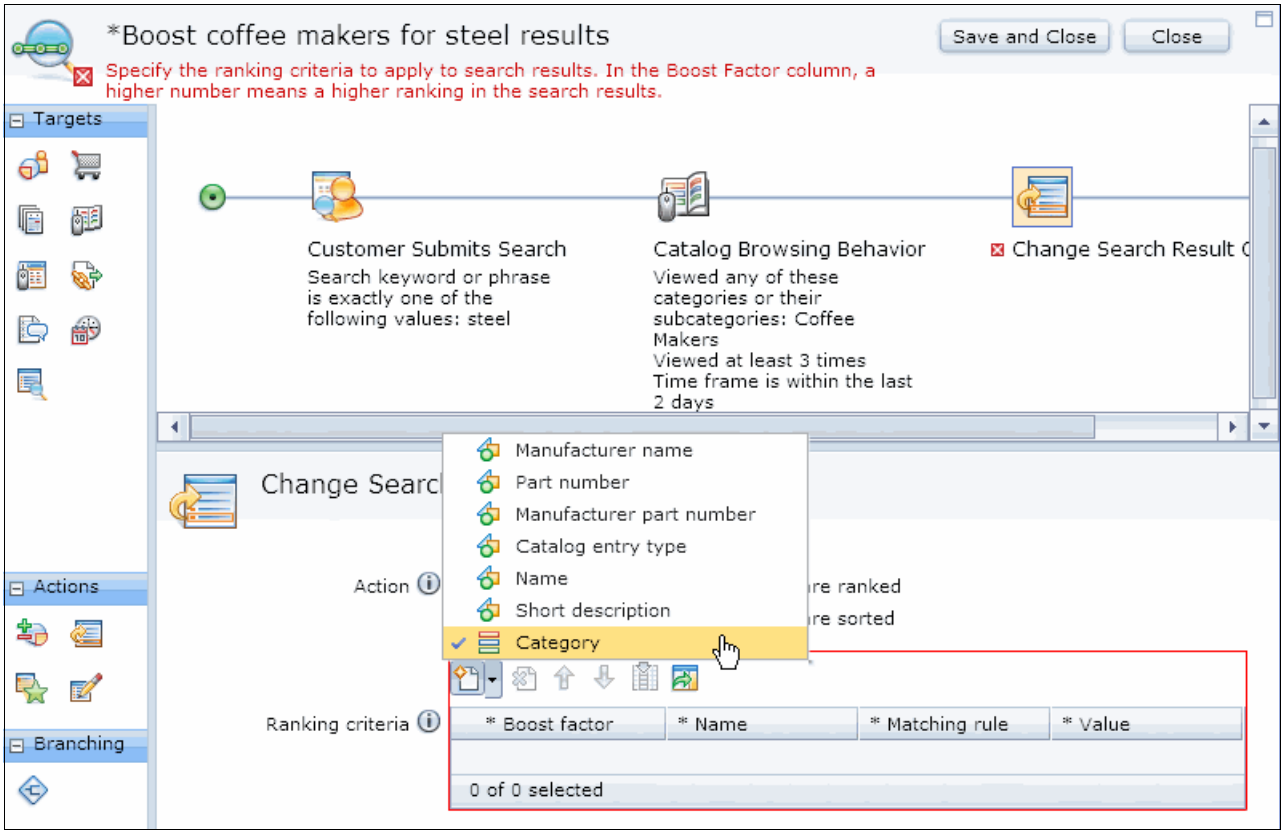


Figure 3-37 Select the ranking criteria for the action

      ii. In the Find and Add pop-up window, enter coffee makers and click **OK**, as shown in Figure 3-38.

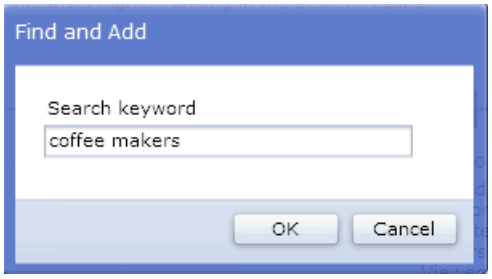


Figure 3-38 Find and add the category

      iii. Enter 25 for the Boost factor.

iv. For the Matching rule, select **Matches**, as shown in Figure 3-39.

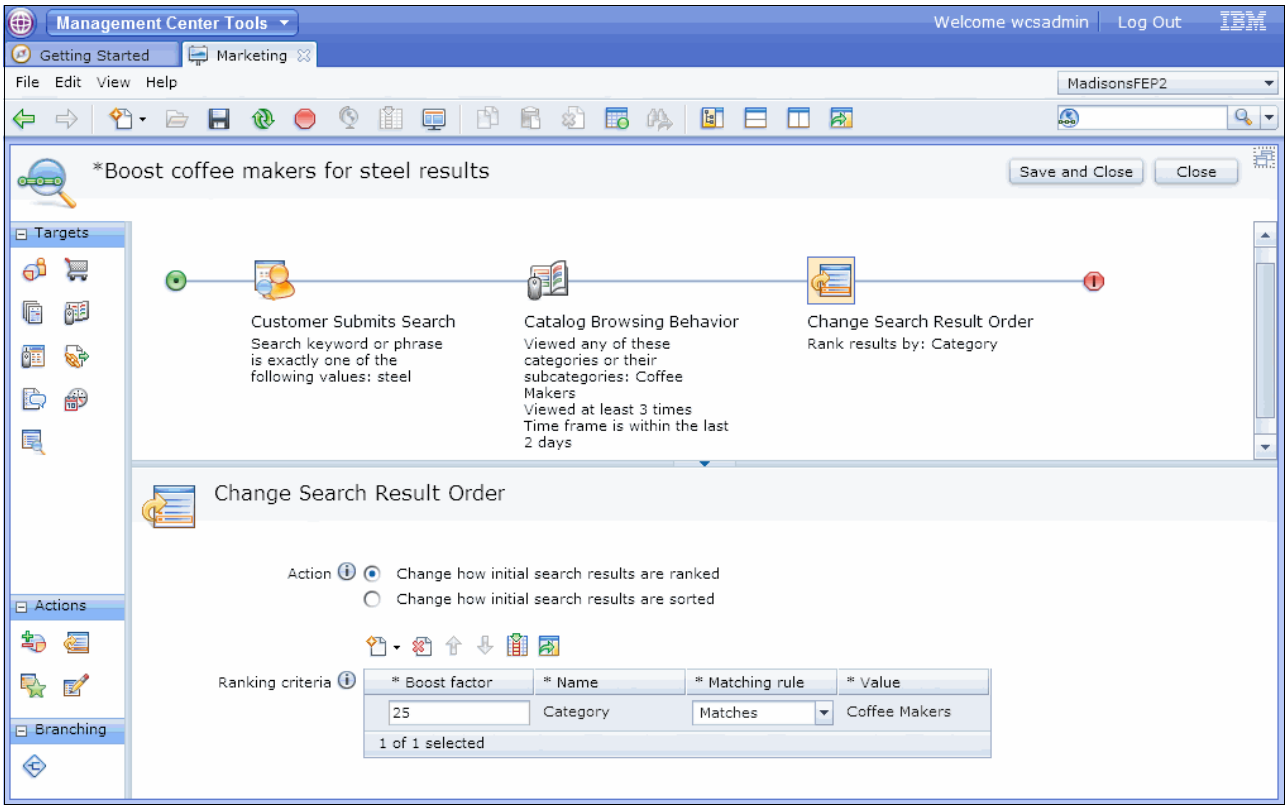


Figure 3-39 Properties for the search rule action

9. Click **Save and Close**.
10. Right-click the current rule from the Search rules list view, and select **Activate**.

11. In the storefront, log in as a user, view the coffee makers category three times, and then search for `steel`. The search results show the products from the coffee makers category boosted to the top, as shown in Figure 3-40.

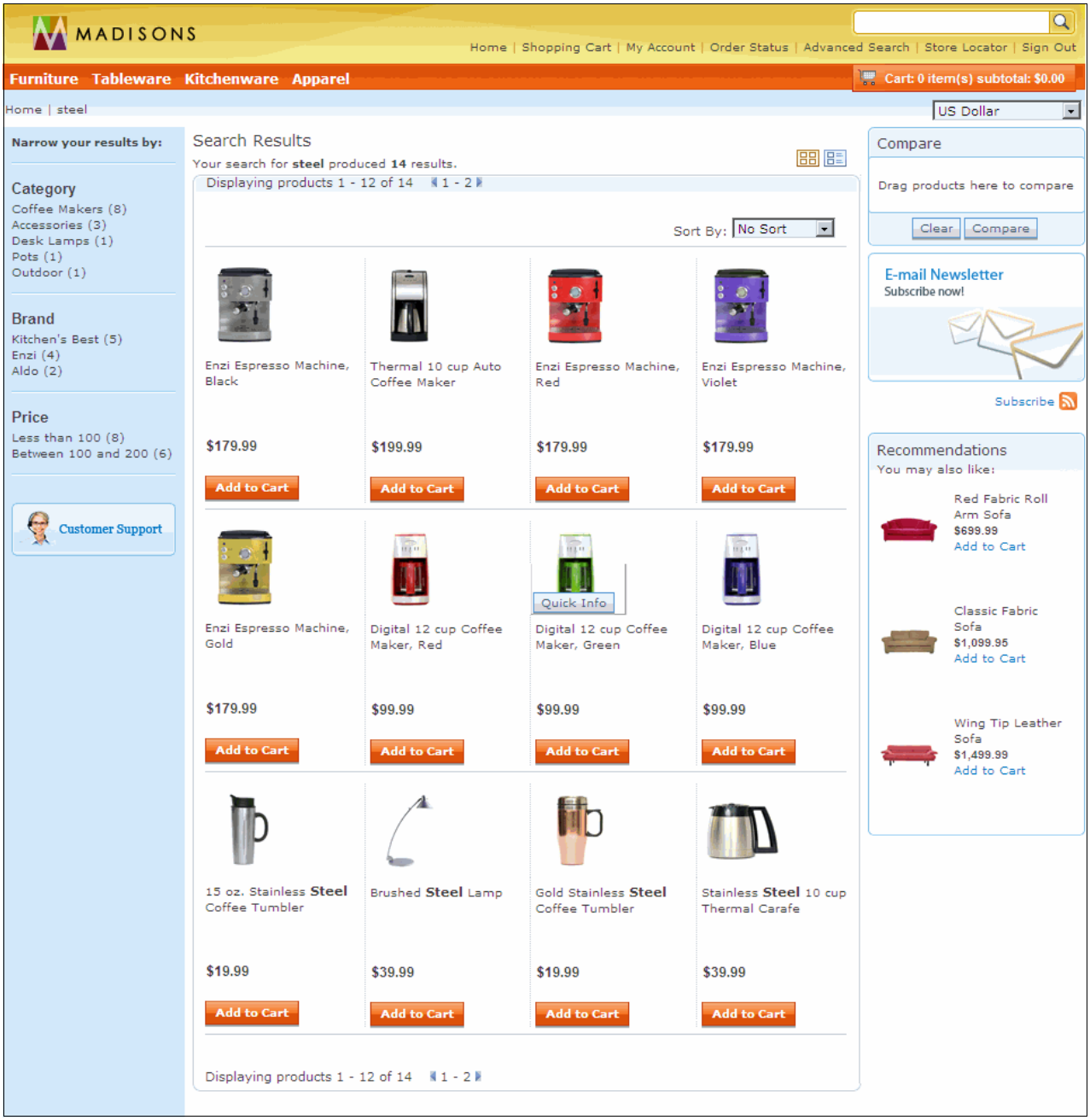


Figure 3-40 Search rule-based search results for steel

3.2.2 Specify Top Search Result

You use the Specify Top Search Result action in a search rule to elevate specific catalog entries to display at the top of the search results list. This action also allows you to specify more than one catalog entry as top search results. Add the catalog entries to the Catalog entries table in the order that you want them to be listed at the top of the search results.

When a customer submits a search, if all the catalog entries in the table have equal relevance to the customer's search, the order that you specify in the table is used in the search results.

When a customer submits a search, if certain catalog entries in the table are more relevant to the customer's search, those catalog entries are listed first in the search results, regardless of the order that you specify in the table.

When shoppers select specific catalog entries to exclude from the search results, those specific catalog entries do not appear in the search results, even if they are specified in this action.

You can configure the action to display products, items, or both. If your default store behavior is to exclude items from the search result, the items configured in this action are not displayed.

In rare cases, the catalog entries that you have configured might not appear at the top of the search result. In this case, you must change the default boost factor for the top search results.

In the current scenario, when a customer searches for `table`, the search results display the product `Breakfast table` with the code `FUOD-O4-BOY-OFF172800-FBON-RSN-QOHO` as the first product in the results.

We show the search results before creating the search rule in Figure 3-41 on page 54.

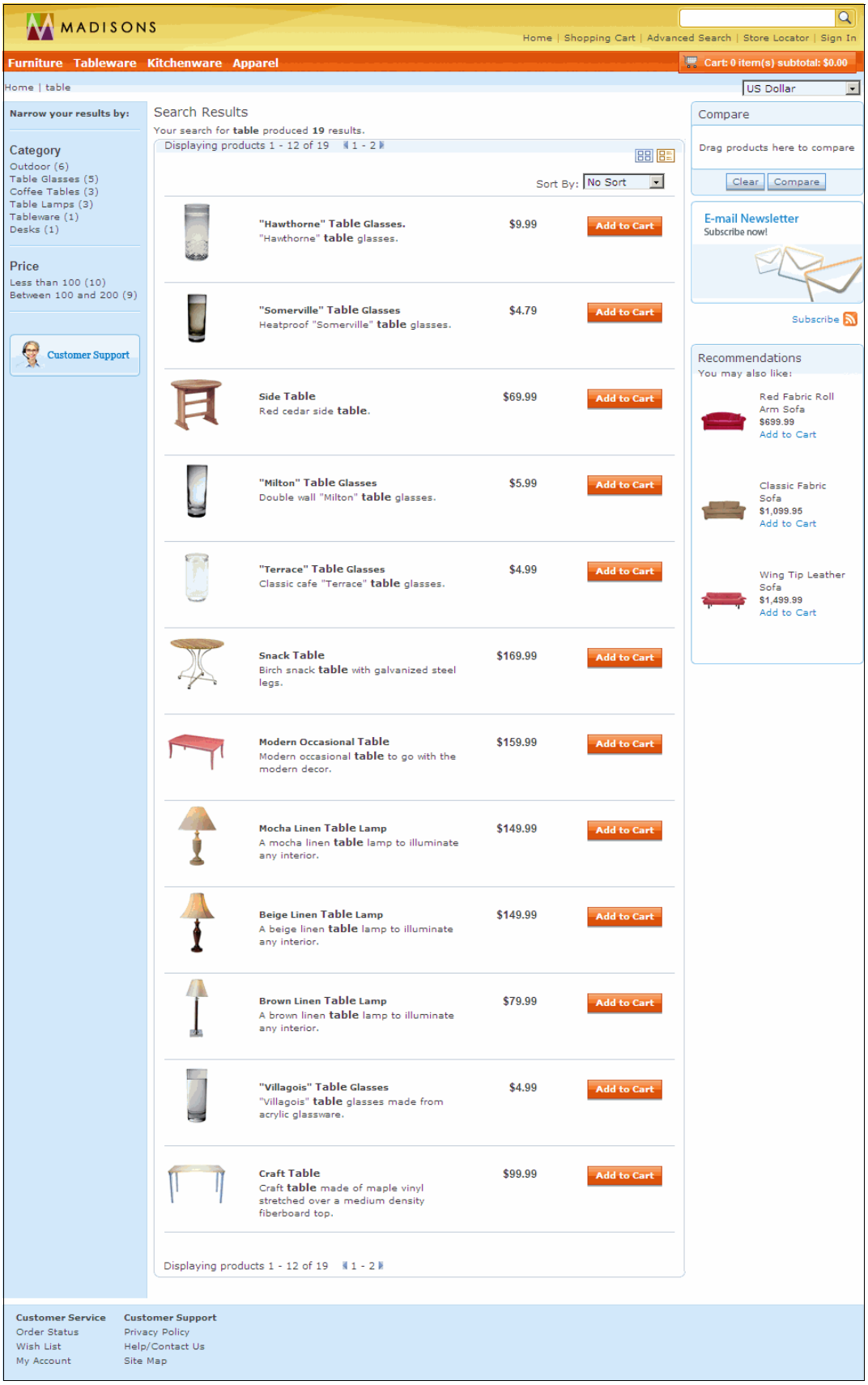


Figure 3-41 Search results for table before implementing the search rule

Follow these steps to implement this scenario:

1. Log on to the Management Center and open the **Marketing** tool.
2. Select **Search Rule** from the Create New toolbar icon.
3. Select the **Specify Top Search Results** template in the pop-up window and click **OK** as shown in Figure 3-42.

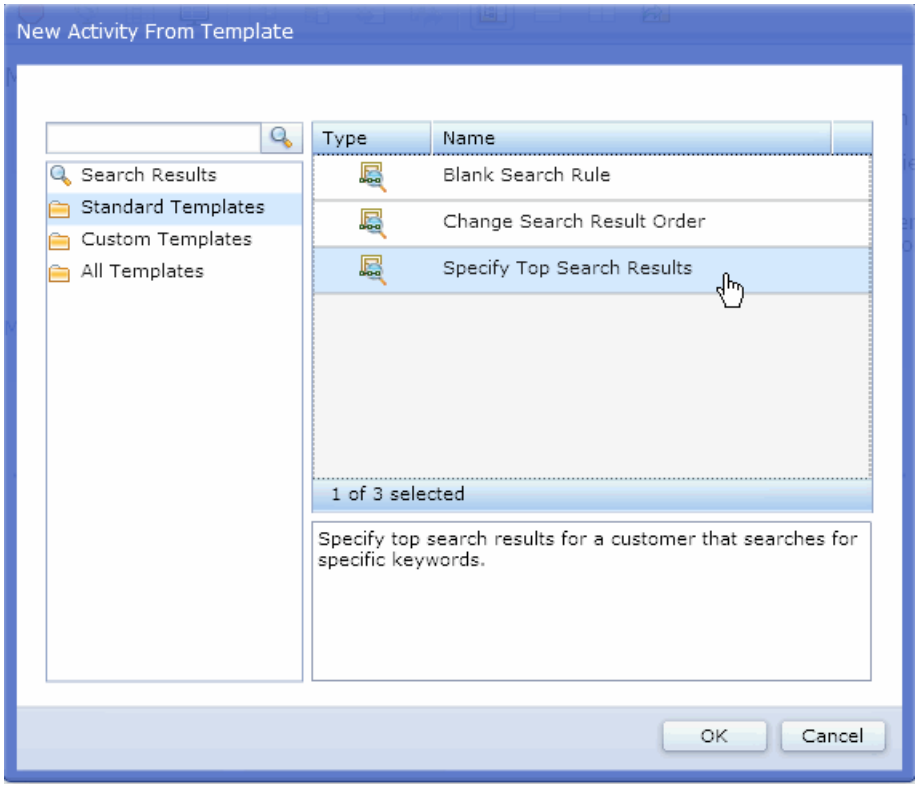


Figure 3-42 Select Specify Top Search Results

4. On the General Properties tab, enter the properties for the search rule, as shown in Figure 3-43.

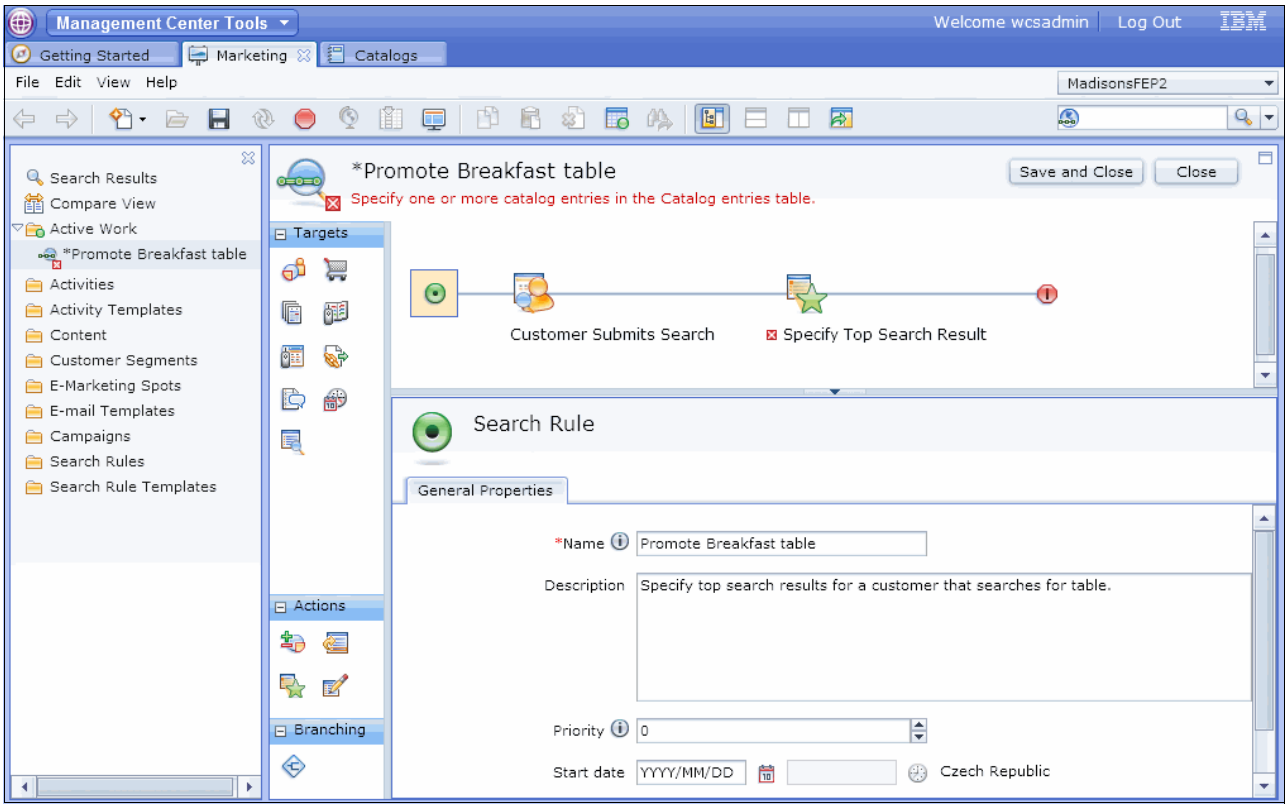


Figure 3-43 Enter Search Rule name

5. Click the **Customer Submits Search** trigger in the work area and enter the following values, as shown in Figure 3-44:

   a. For Matching rule, select **Search phrase contains one of the following words**.

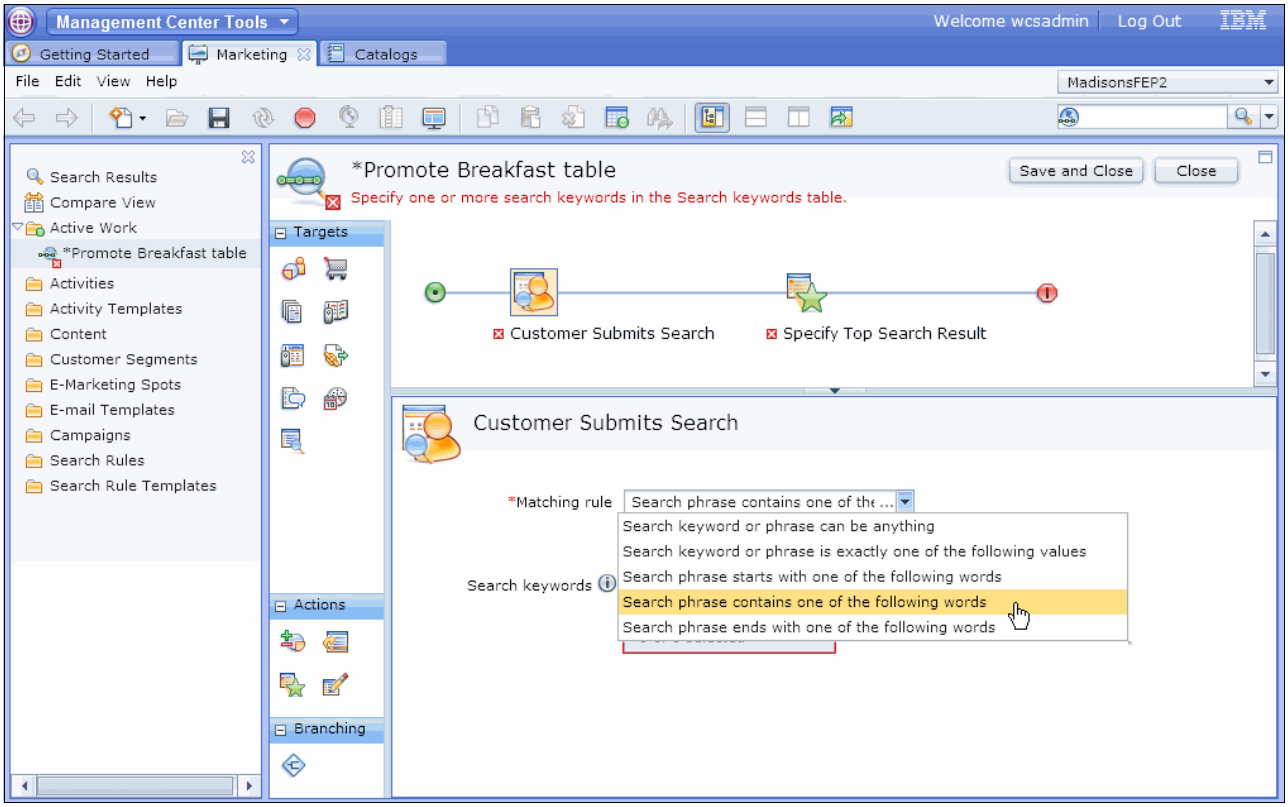


Figure 3-44 Select Matching rule

   b. Click **Create New Keyword** and add `table` as a keyword.

6. Click **Specify Top Search Result** in the work area and enter the catalog entry ID for the Breakfast table, `FUOD-04-BOY-OFF172800-FBON-RSN-QOH0`, in the input text box. Click **Find and Add**, as shown in Figure 3-45.

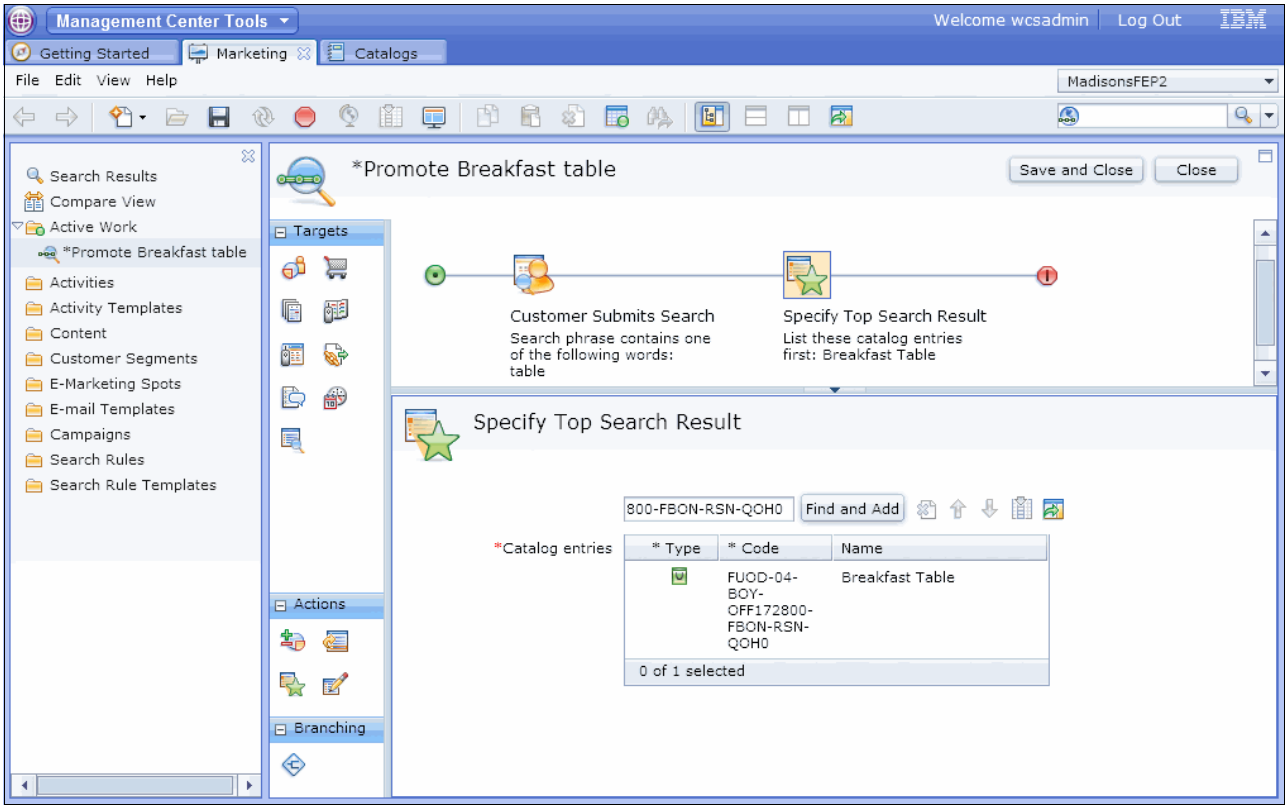


Figure 3-45 Find and add the catalog entry to be specified as the top result in the search results

7. Click **Save and Close**.

8. The list of search rules is displayed. Right-click the current search rule and click **Activate**, as shown in Figure 3-46.

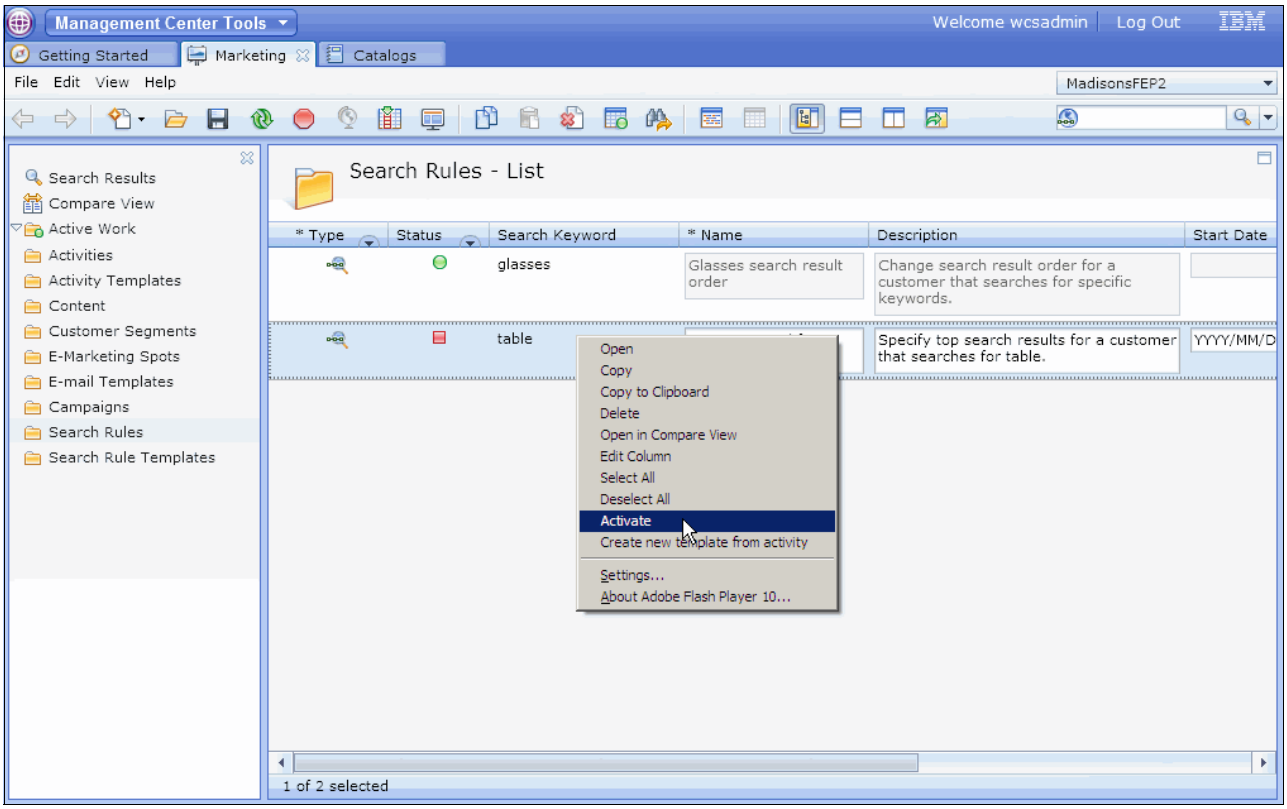


Figure 3-46 Activate Search Rule

9. Search for `table` in the storefront. Breakfast table is the first search result, as shown in Figure 3-47 on page 60.

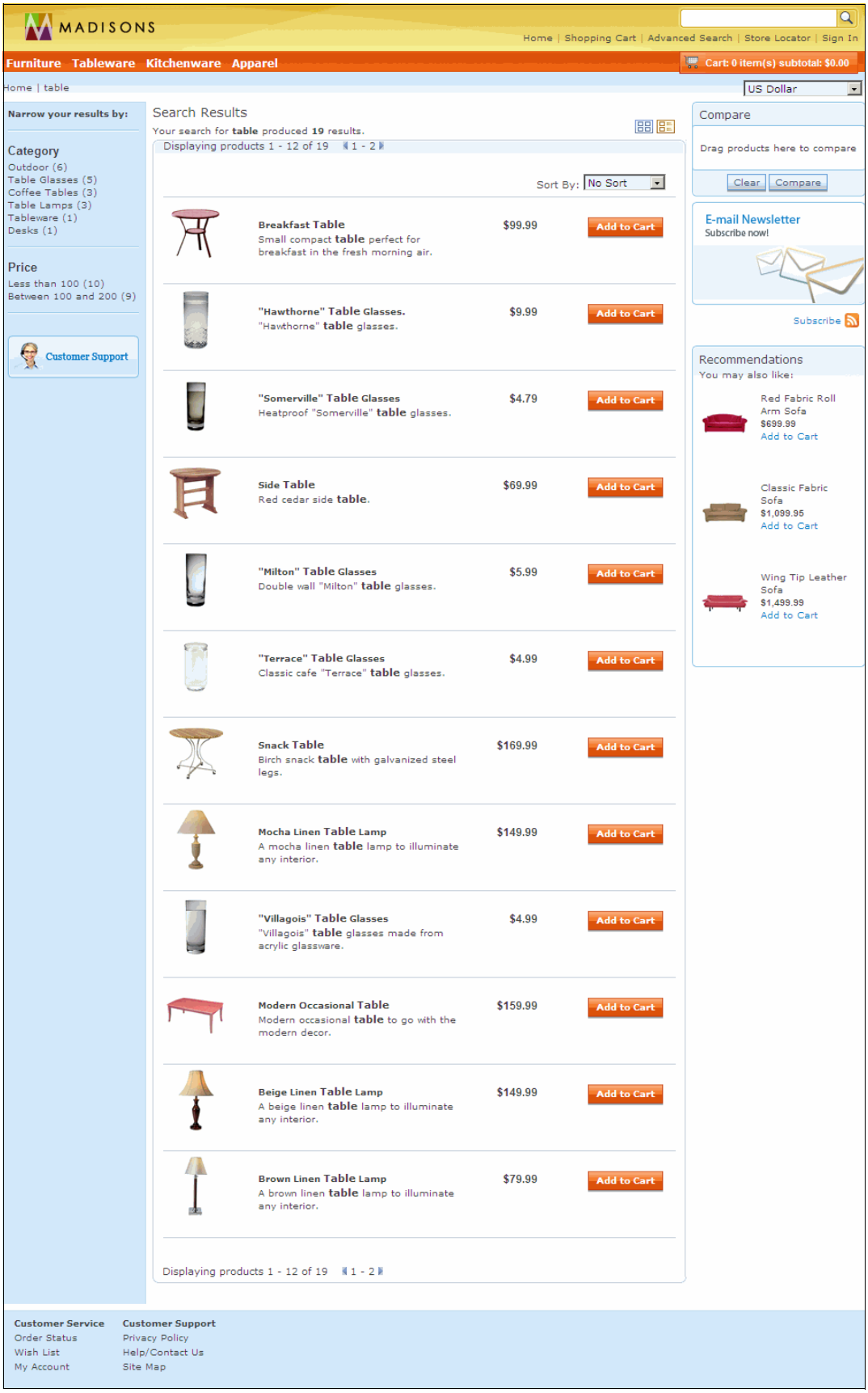


Figure 3-47 Search rule-based search results

3.2.3 Add or replace search criteria

You can use this action if you need to add or replace the search criteria when customers submit searches in the store. By adding search criteria, you can filter the search results further. You can modify the search results by replacing the search criteria.

Search term associations: If you use this action to replace search terms and your search rule does not have any targets, you might consider creating search term associations instead.

In the current scenario, a shopper is searching for a table. That shopper has purchased products under the Furniture category within the most recent 10 orders. The system needs to restrict the search results to the Furniture category and filter out products from other categories, such as `table glasses`. Perform the following steps to implement this scenario:

1. Log on to the Management Center and open the **Marketing** tool.
2. Select the **Search Rule** from the Create New toolbar icon.
3. Select the **Blank Search Rule** template in the pop-up window and click **OK**, as shown in Figure 3-48.

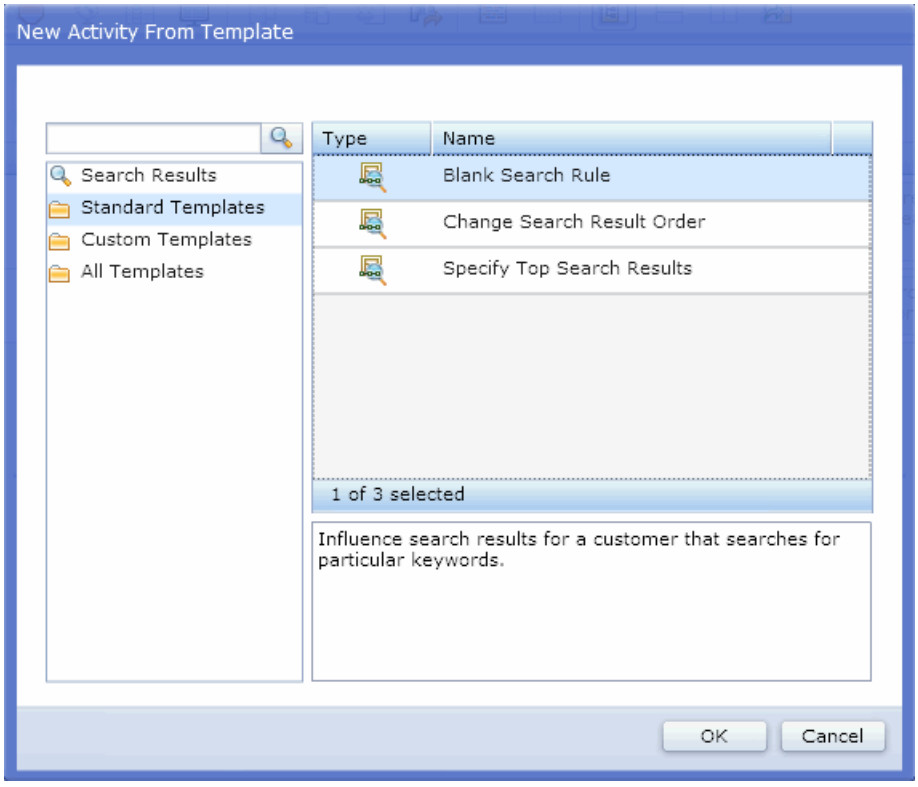


Figure 3-48 Select the Blank Search Rule template

4. Enter the properties of the search rule on the General Properties tab, as shown in Figure 3-49.

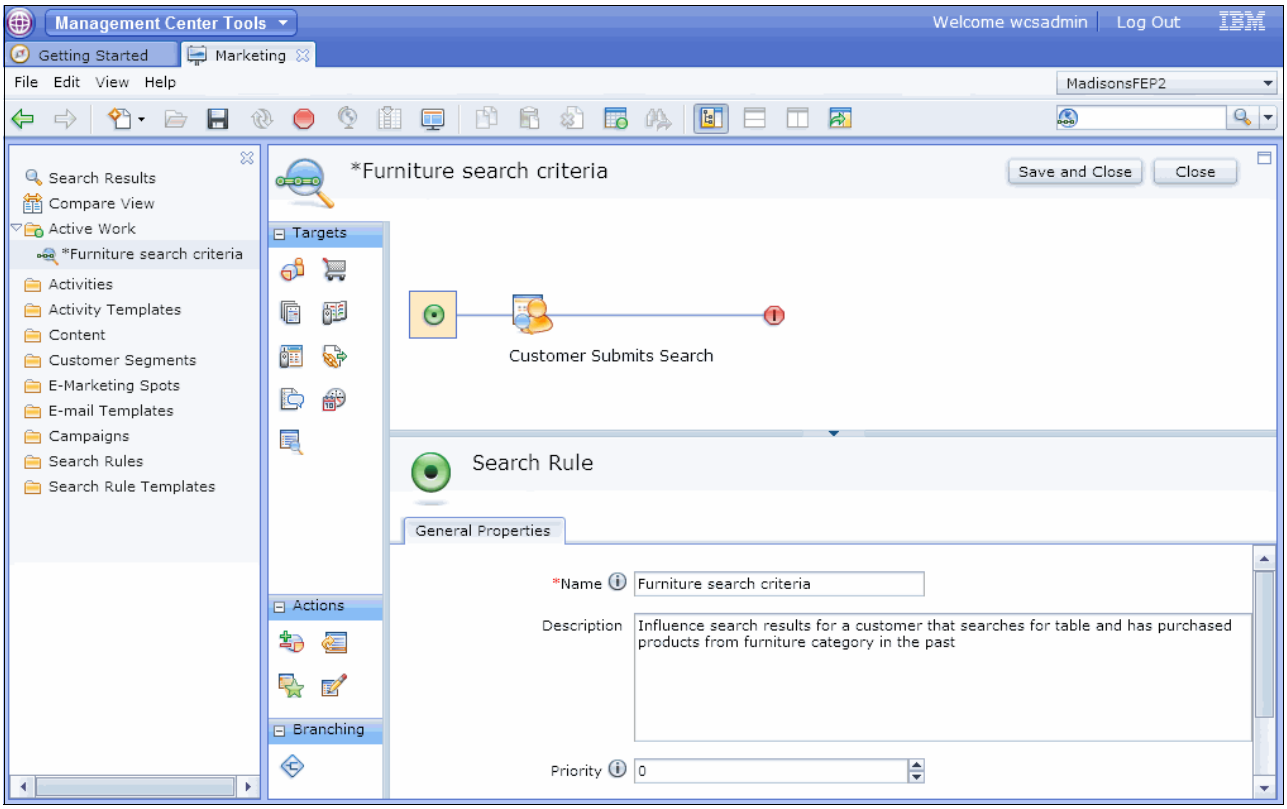


Figure 3-49 Enter Search Rule name

5. Click the **Customer Submits Search** trigger in work area and enter the following values, as shown in Figure 3-50:
   a. Select the matching rule **Search keyword or phrase is exactly one of the following values** in the properties view.
   b. Add the search keyword `table`.

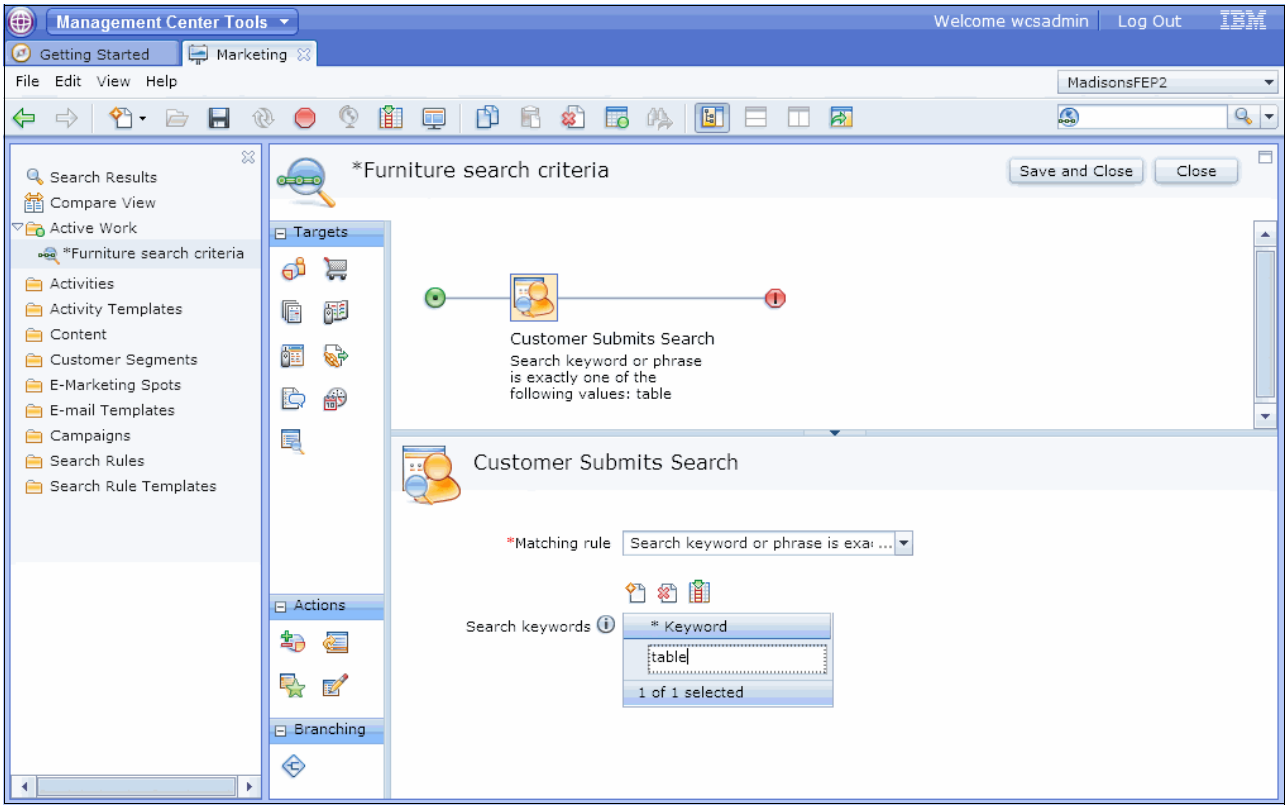


Figure 3-50 Enter the trigger properties

6. Drag and drop the target **Purchase history** into the work area.

7. Click **Purchase History** and enter the following values, as shown in Figure 3-51:
   a. For Purchase history focus, select **Purchases of catalog entries from specific categories**.
   b. For Target customers, select **Who have catalog entries from any of the following categories and satisfy the following conditions**.
   c. For Categories, click **Find and Add** for `furniture`.
   d. For Number of catalog entries, select **At least the following number**.
   e. For Number, select `1`.
   f. For Value of catalog entries, select **Any value**.
   g. For Time frame, select **In the last number of orders**.
   h. For Number of orders, select **10**.

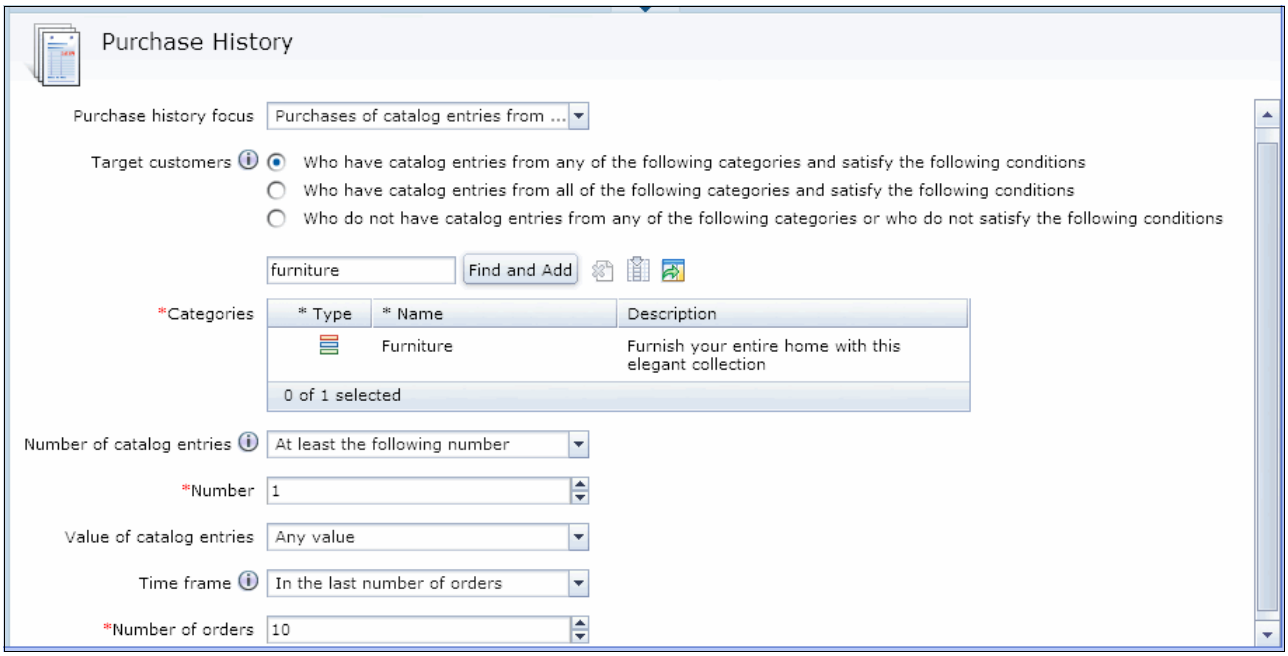


Figure 3-51 Enter Purchase History properties

8. Drag and drop **Add or Replace Search Criteria** into the work area and enter the following values, as shown in Figure 3-52:
   a. For Action, select **Add search criteria**.
   b. For Search filters, select Category **Matches** Furniture.

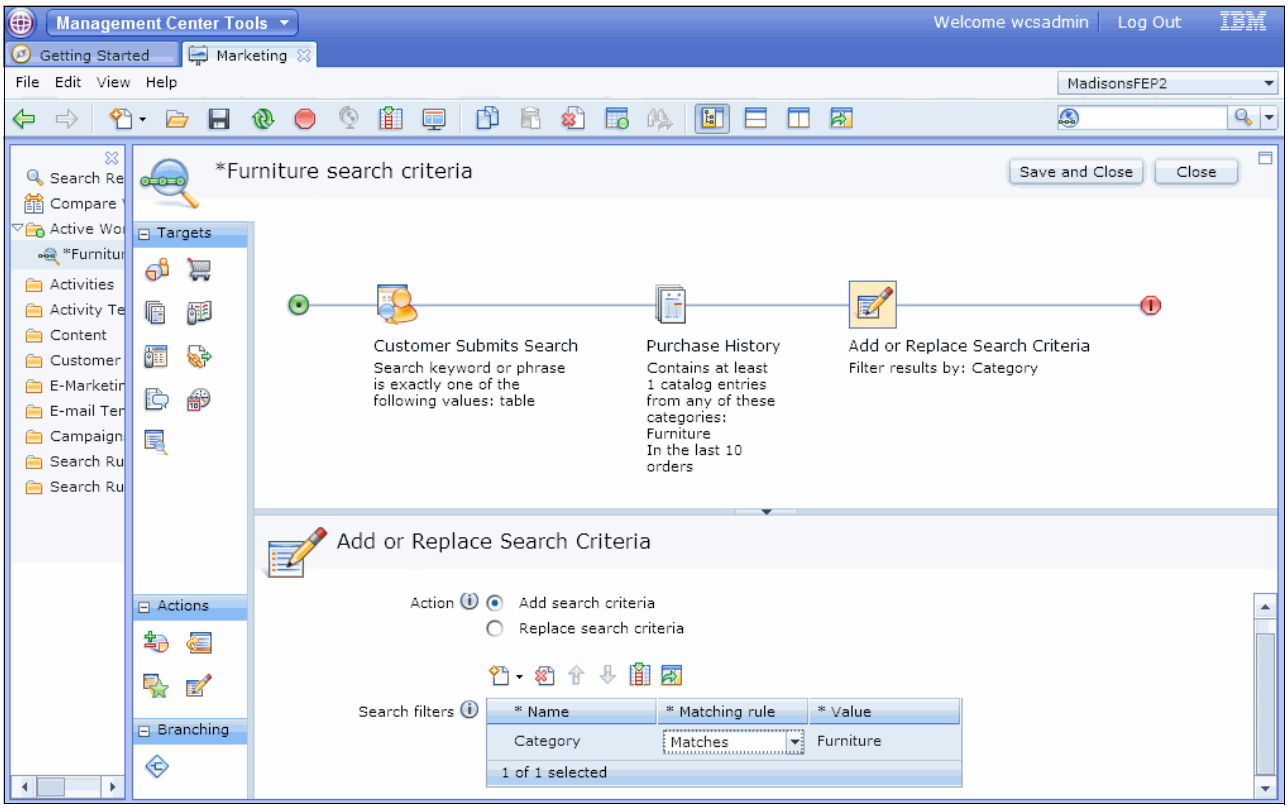


Figure 3-52 Enter the properties for the action

9. Click **Save and Close**.

10. Right-click the search rule from the Search Rules - List and click **Activate**, as shown in Figure 3-53.

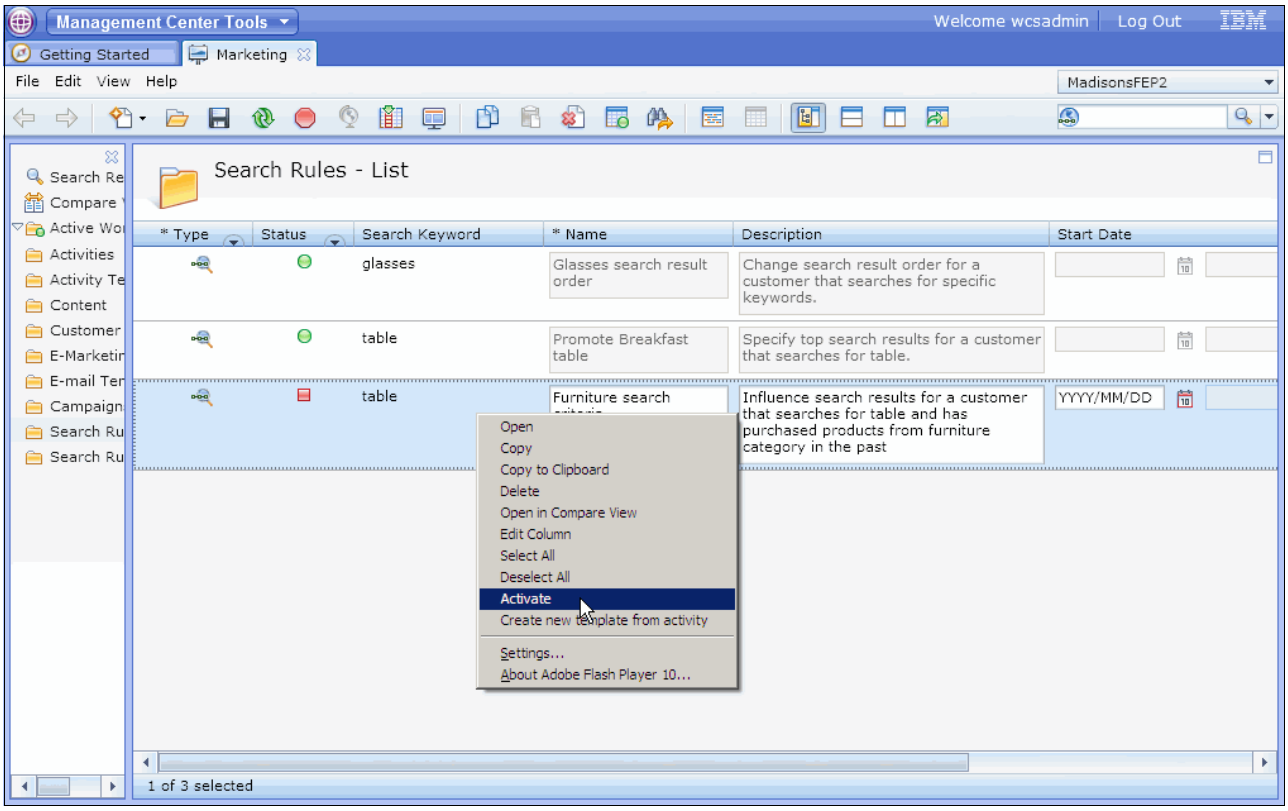


Figure 3-53 Activate the search rule

11. Log in to the store as a user who has previously placed an order that included products from the Furniture category. Figure 3-54 shows the order history of a user who purchased a product from the Furniture category.

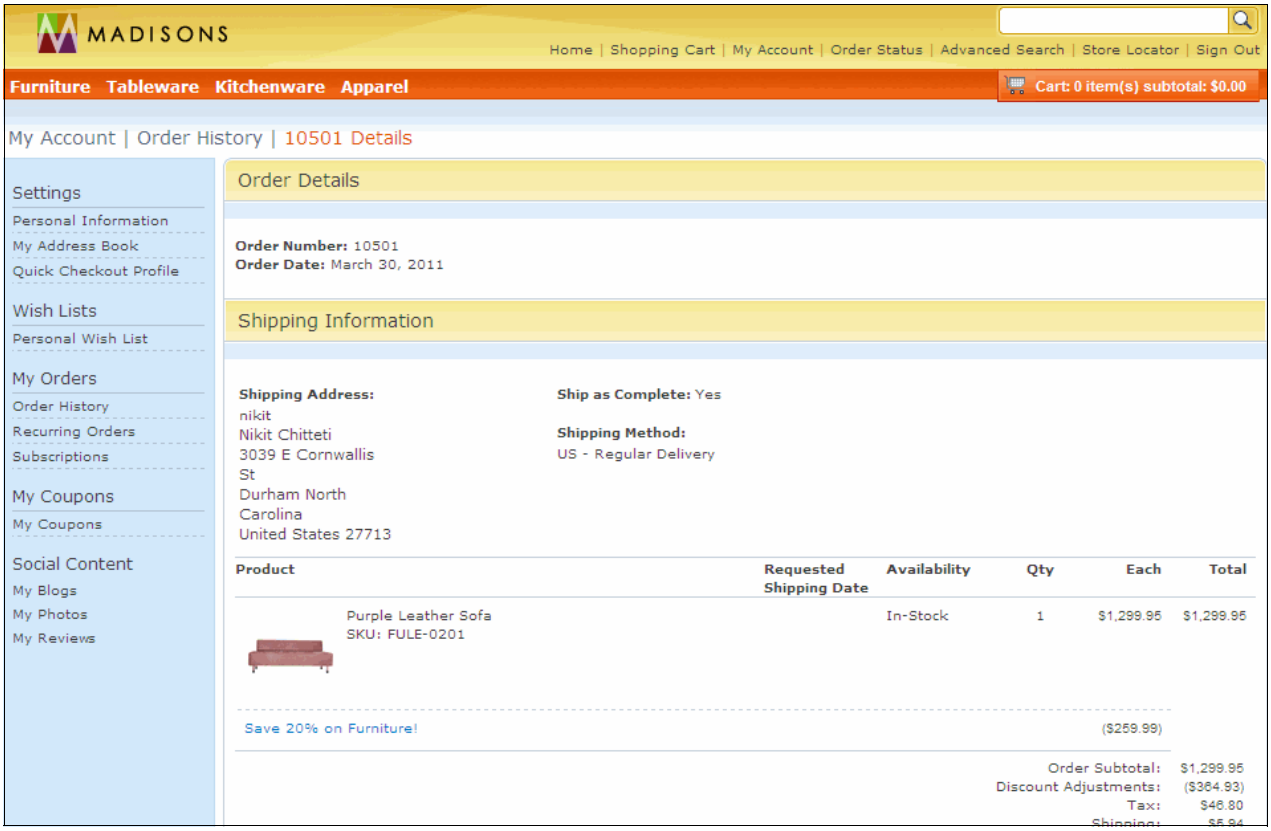


Figure 3-54 Order details with one product in the furniture category

12. Search for `table`. The search results do not contain the products from the `tableware` and `table Glasses` categories, as shown in Figure 3-55.

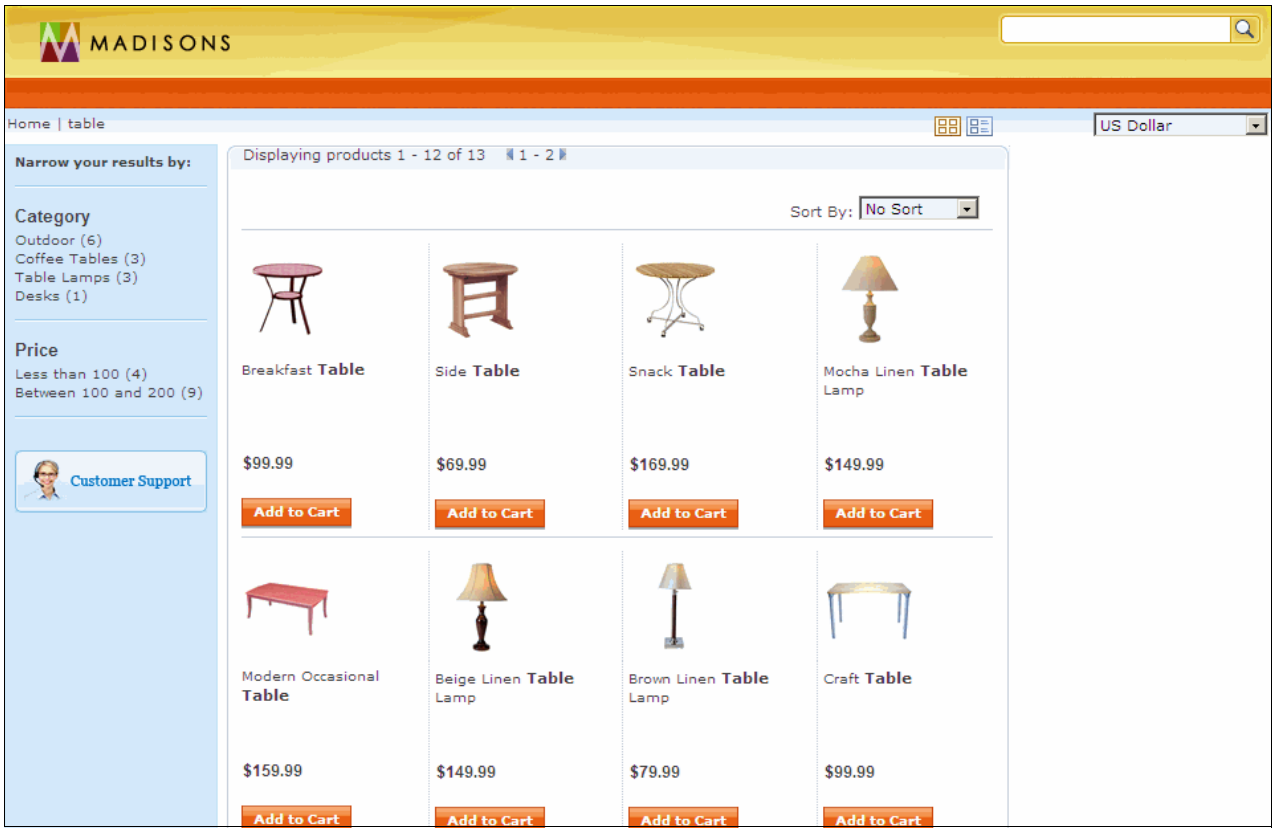


Figure 3-55 Results for table after the search rule was enabled (observe the facet category)

3.2.4 Product recommendations

You can use the search-based activity of the shoppers to target customers for marketing activities. In the current scenario, when shoppers search for `cup`, the system needs to recommend the following products: `Digital 12 cup coffee maker, Blue` with SKU: `KB-04B` and `Coffee and Espresso Bar` with SKU: `KIES-01`. To implement this product recommendation process, you must create a new web activity that targets customers based on their search criteria and results. Follow these steps to these implement product recommendations:

1. Log on to the Management Center and open the **Marketing** tool.

2. From the **Create New** toolbar icon, select **Web Activity**, as shown in Figure 3-56.

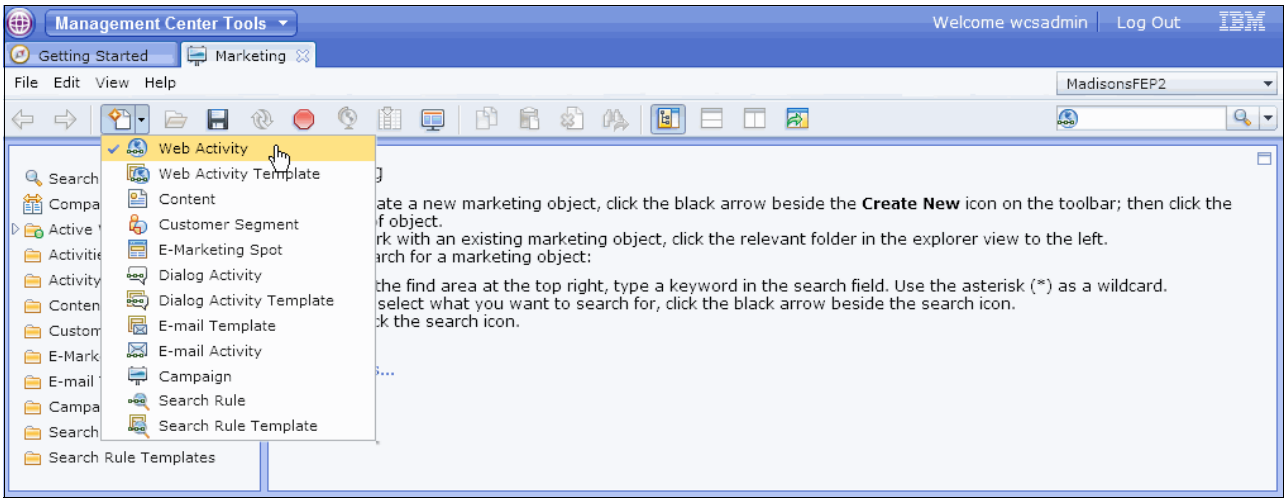


Figure 3-56 Create Web Activity

3. Select the **Catalog Entry Recommendation** template and click **OK**, as shown in Figure 3-57.

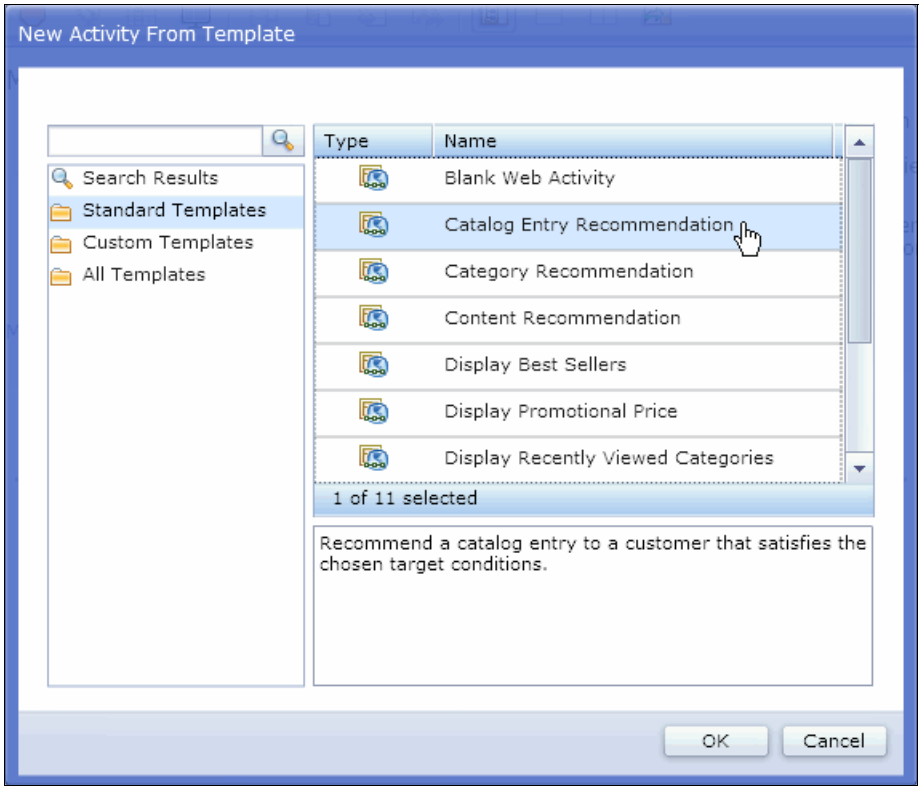


Figure 3-57 Select the Web activity template

4. Enter the following values, as shown in Figure 3-58, and keep all other default values:
   a. For Name, enter `Recommendations for cup`.
   b. For Priority, select **100**.
   c. For Start date, select **Today**.

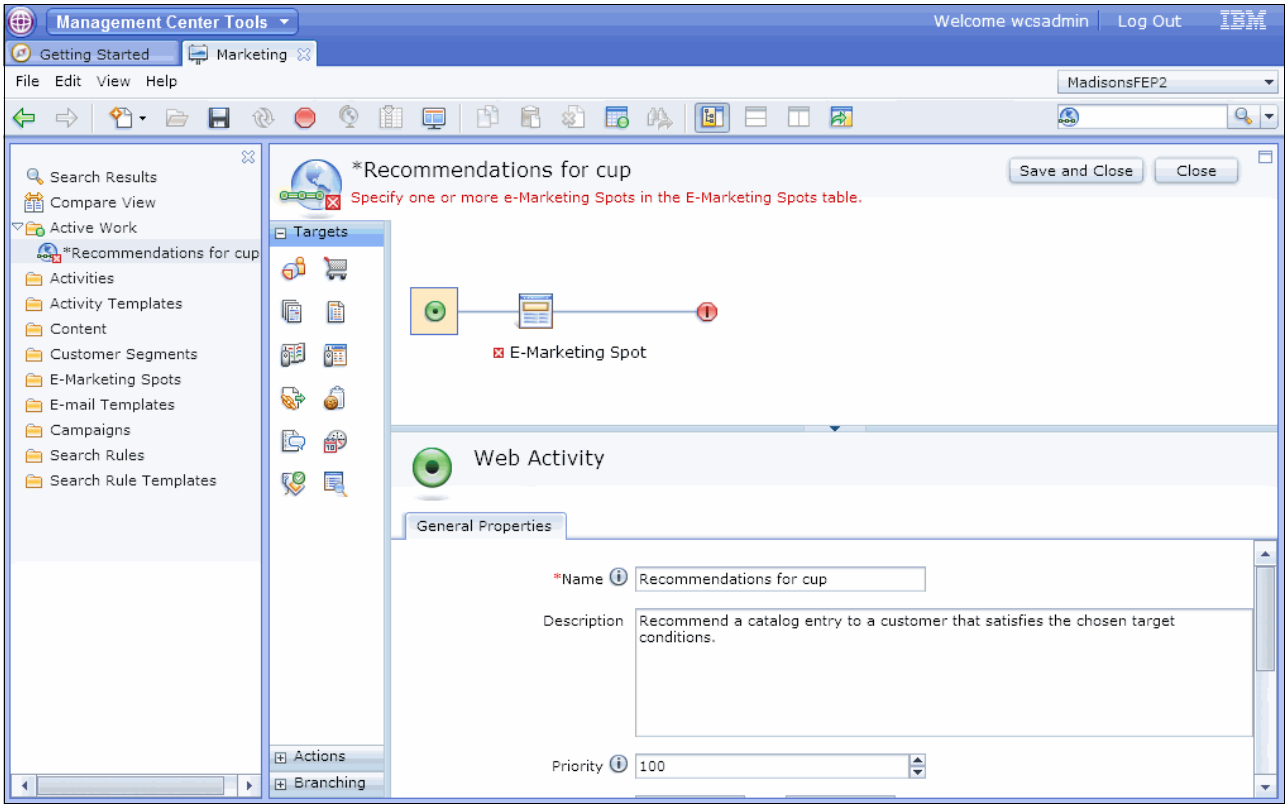


Figure 3-58 Enter Web Activity properties

5. Select **E-Marketing Spot**.

6. Find and add the **RightSideBarFeaturedProducts** E-Marketing Spot (ESpot), as shown in Figure 3-59. In this ESpot, we want to show the recommended products in the rightmost sidebar on the search results page.

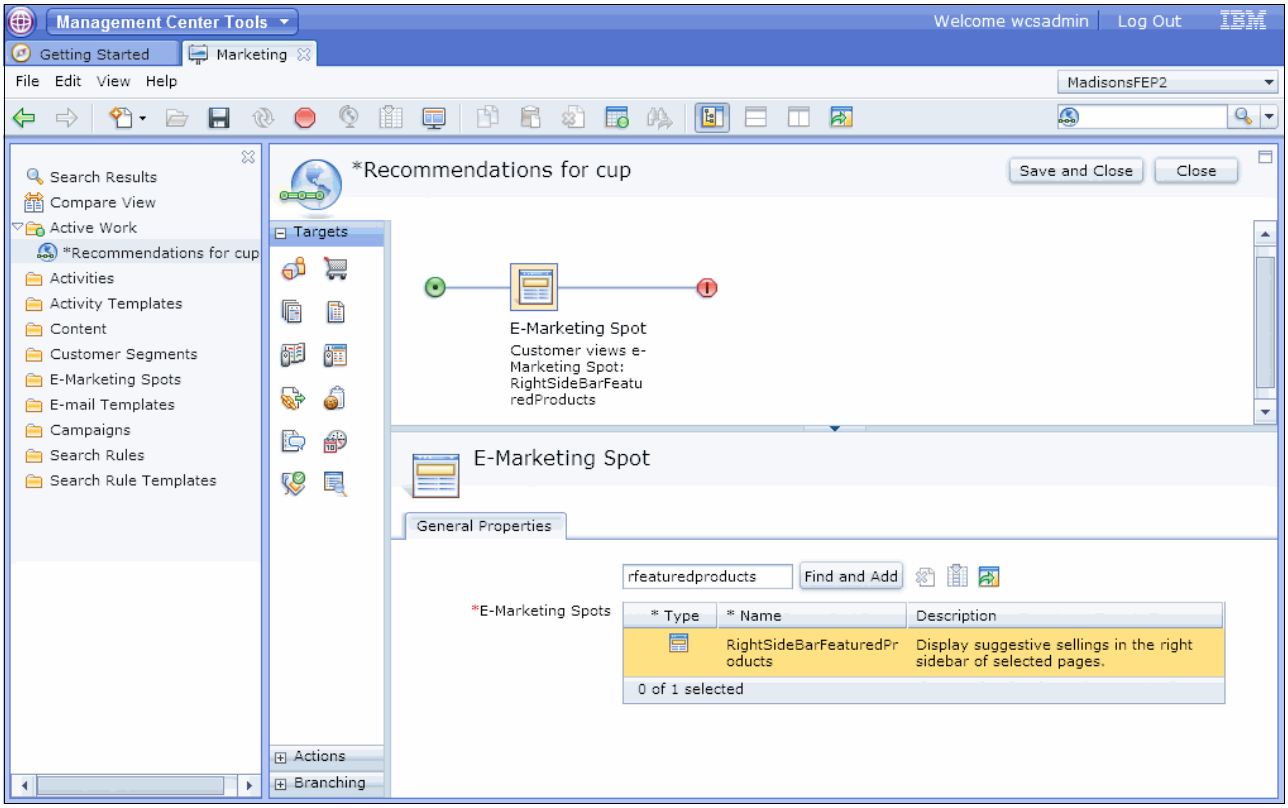


Figure 3-59 Select the E-Marketing Spot to show the recommendations

7. Drag and drop the **Current Page** target into the work area and enter the following values, as shown in Figure 3-60, Figure 3-61 on page 73, and Figure 3-62 on page 74:

   a. For Customer behavior, select **Customer is viewing search results**.

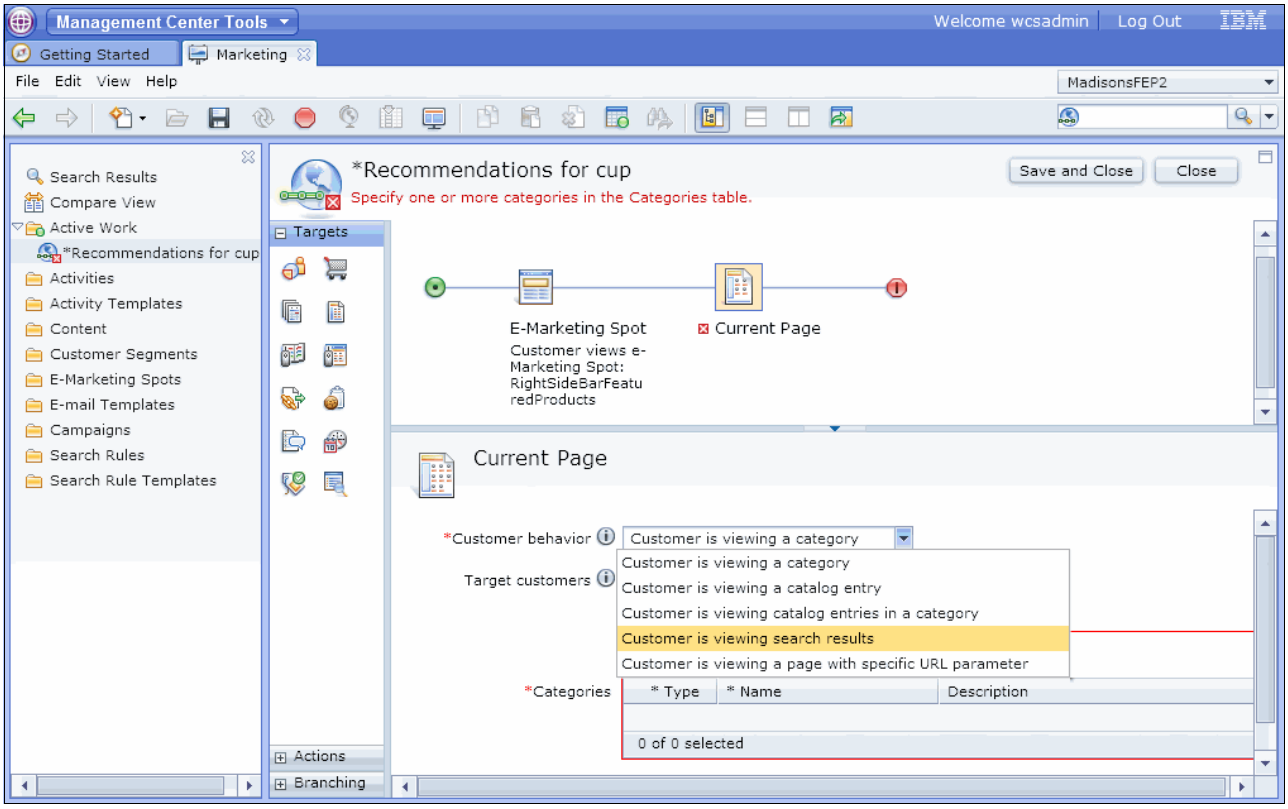


Figure 3-60 Select the Customer behavior

   b. For Target customers, select **Who are searching for any of the following keywords**.

   c. For the Keyword matching rule, select **Search keyword is exactly one of the following values**.

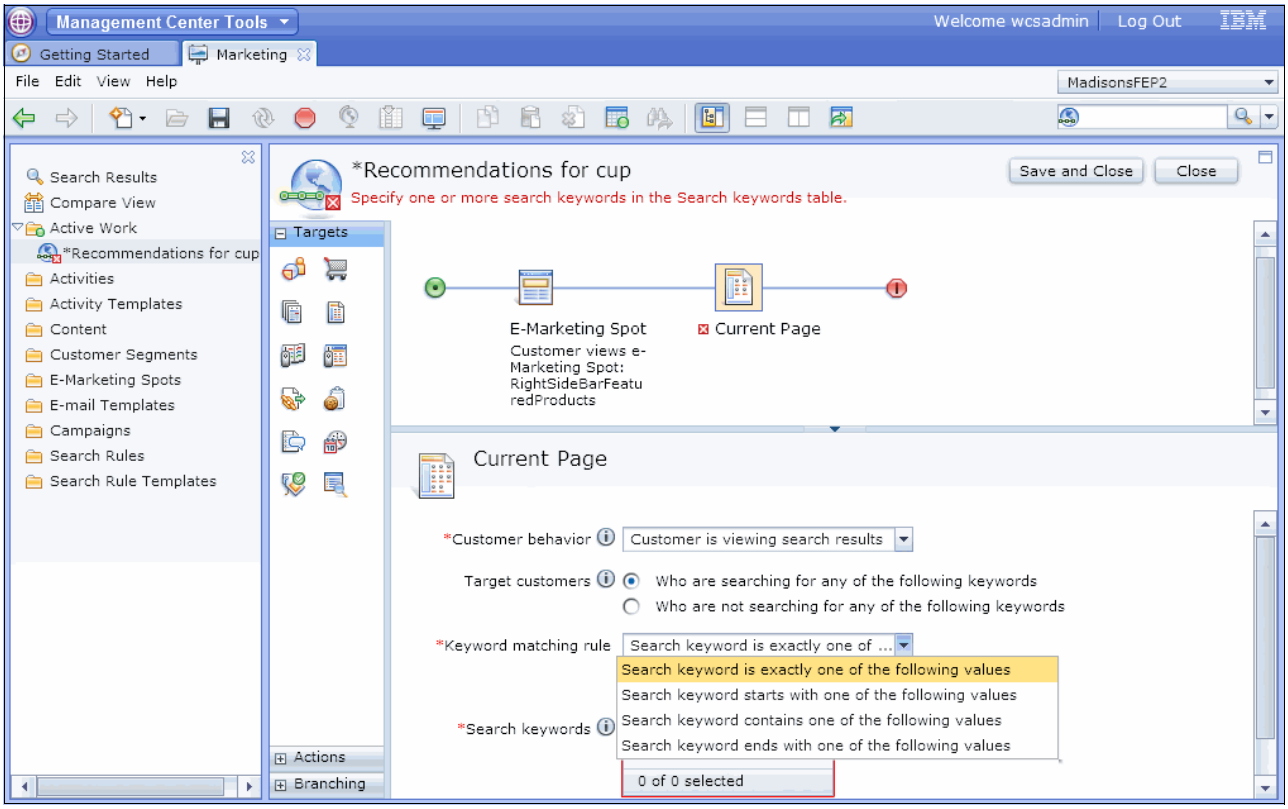


Figure 3-61 Select keyword matching rule

d. For Search keywords, click the **New** icon and add `cup`.

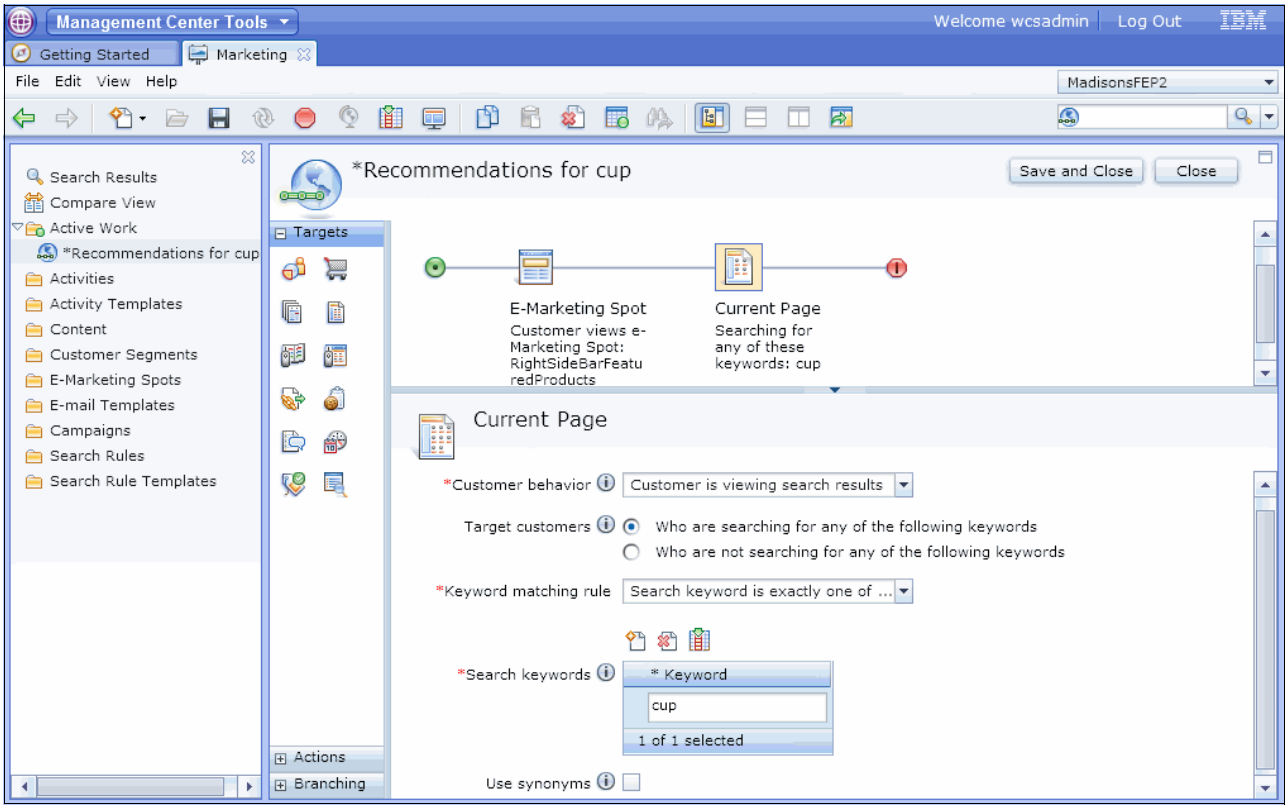


Figure 3-62 Enter the keyword cup

8. Drag and drop the **Recommend Catalog Entry** action into the work area and enter the following values, as shown in Figure 3-63:
   a. For Recommendation method, select **Specify a list of catalog entries**.
   b. For Catalog entries, add the codes `KIES-01` and `KB-04B`.

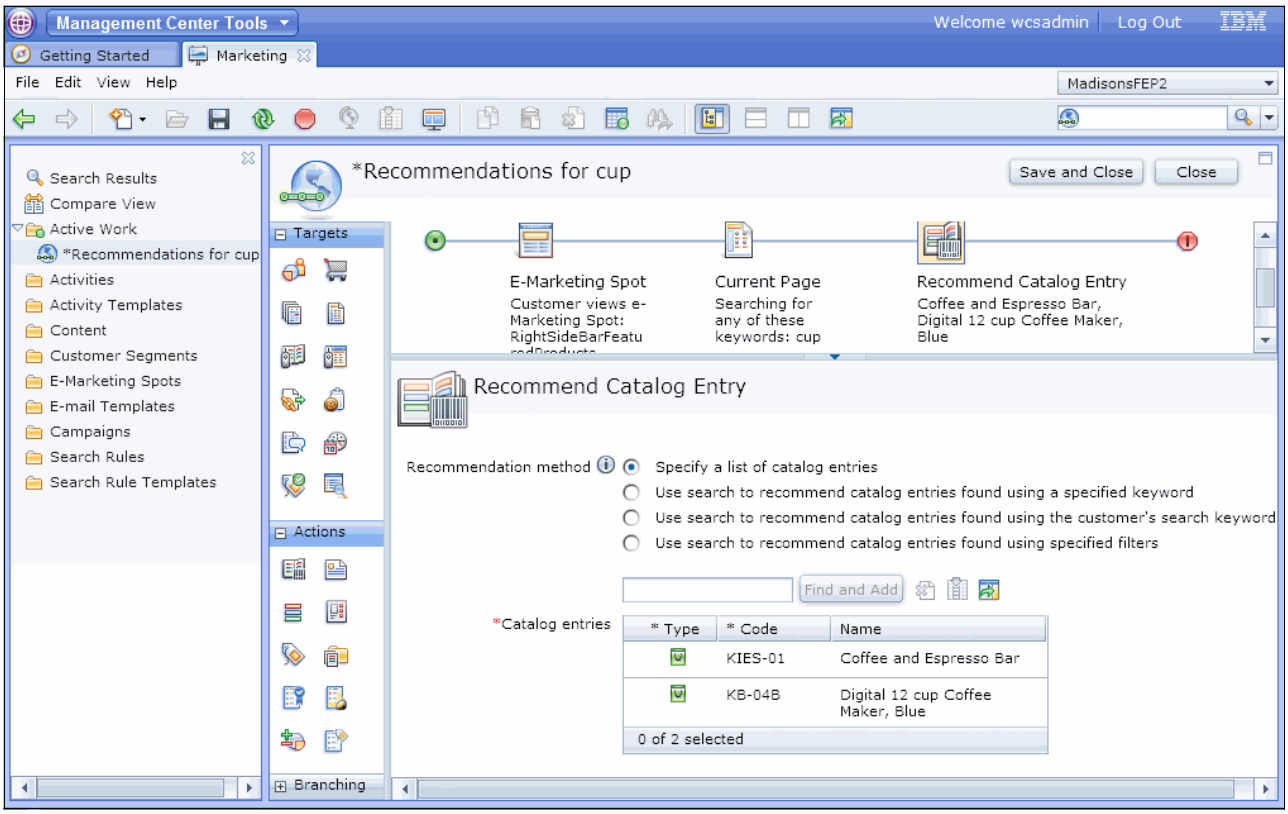


Figure 3-63 Add the recommended catalog entries

9. Click **Save and Close**.

10. Right-click the activity and click **Activate**, as shown in Figure 3-64.

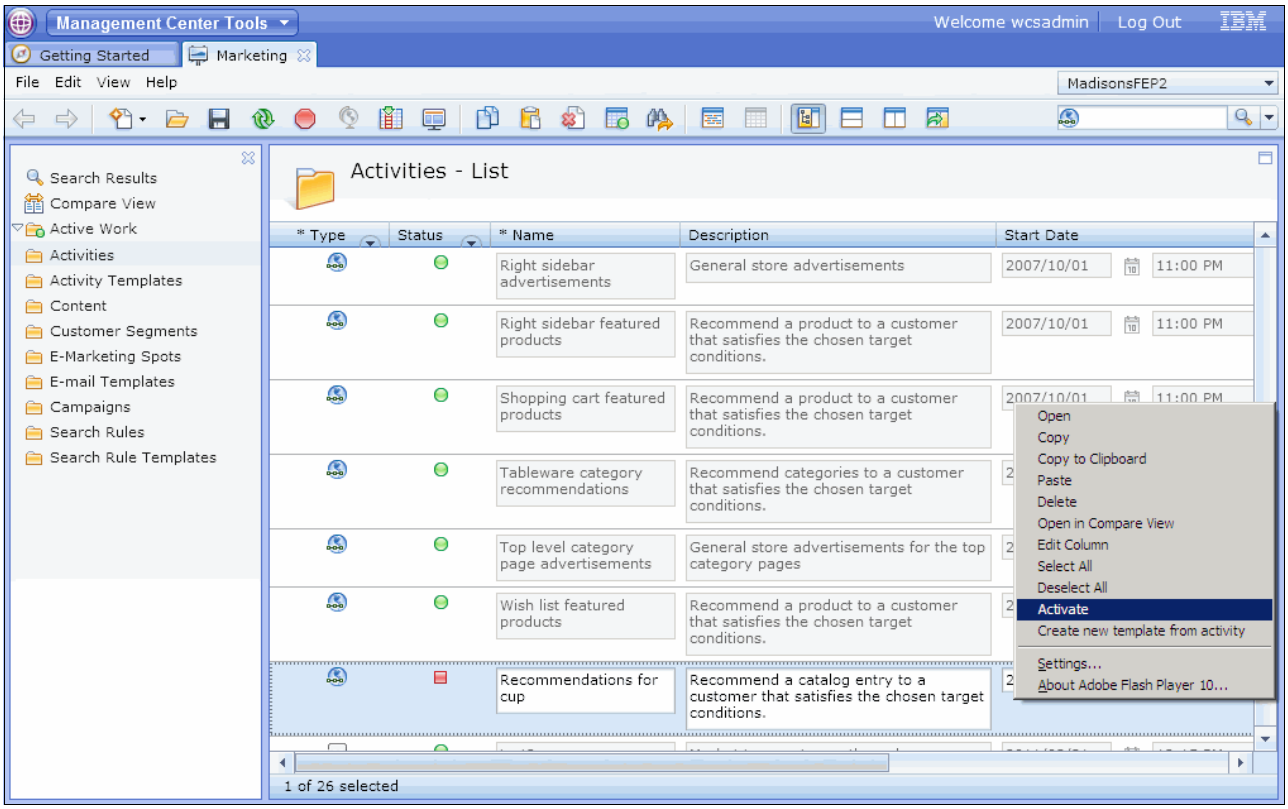


Figure 3-64 Activate the web activity

If a shopper searches for `cup` before activating this rule, the search results contain the default recommendations for products under the furniture category, as shown in Figure 3-65 on page 77.

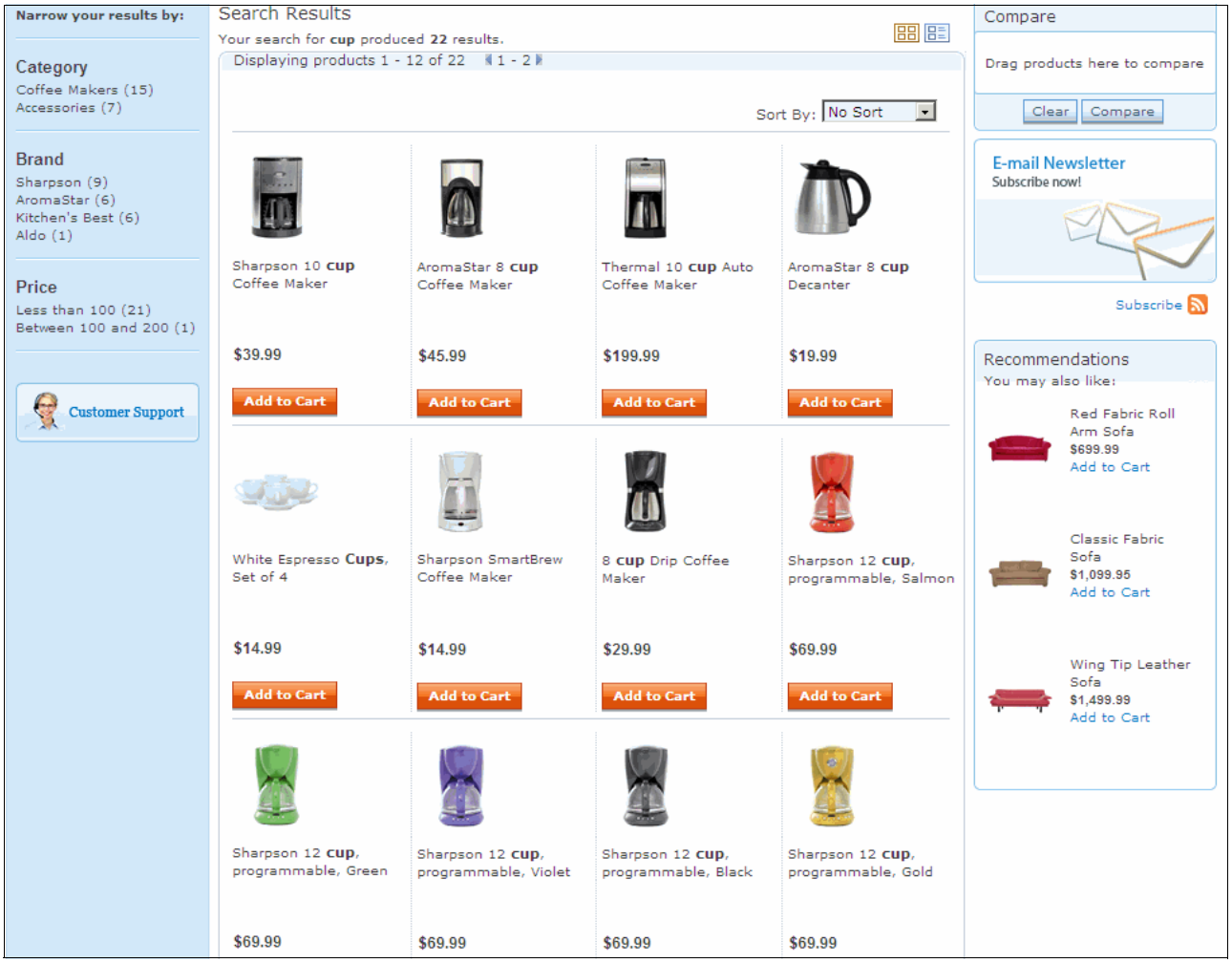


Figure 3-65 Recommendations E-Spot before activating the rule for the search term cup results

After activating this rule, the Search Results page shows the rightmost Recommendations tab with the coffee makers that were specified in the rule, as shown in Figure 3-66 on page 78.

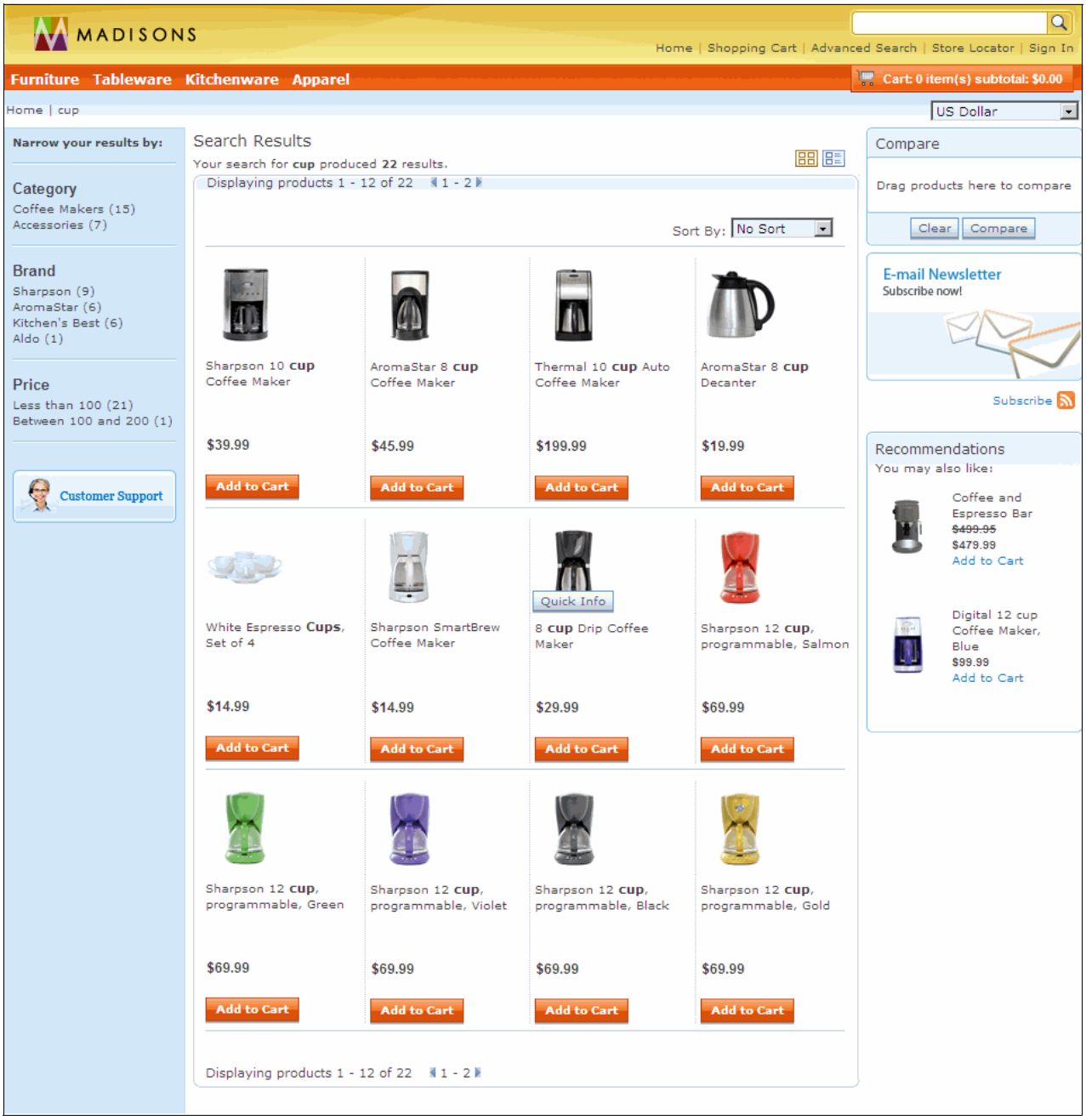


Figure 3-66 Recommendations E-Spot after activating the rule for search term cup results

3.2.5 Product promotion

In the current scenario, the system issues a 15% off of the total order coupon to customers who search for `bodysuit` within the specified time frame. To implement this product promotion, you must create a coupon promotion and it needs to be associated to a new dialog activity that defines the trigger to issue the coupon. You must create a new dialog activity to communicate the promotion information to the customer based on the search trigger.

Follow these steps to create the promotion:

1. Log on to the Management Center.

2. Open the **Promotions** tool, as shown in Figure 3-67.

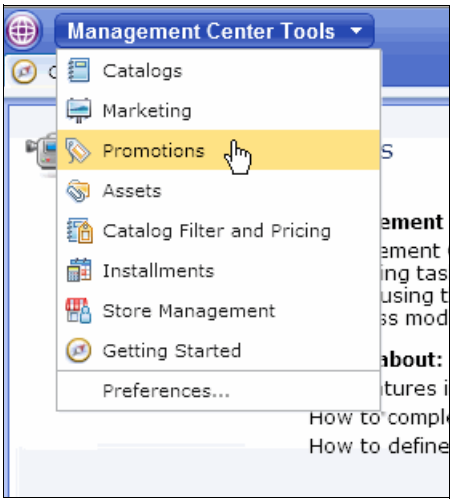


Figure 3-67 Open the Promotions tool

3. Select **Promotion** from the Create New toolbar icon, as shown in Figure 3-68.

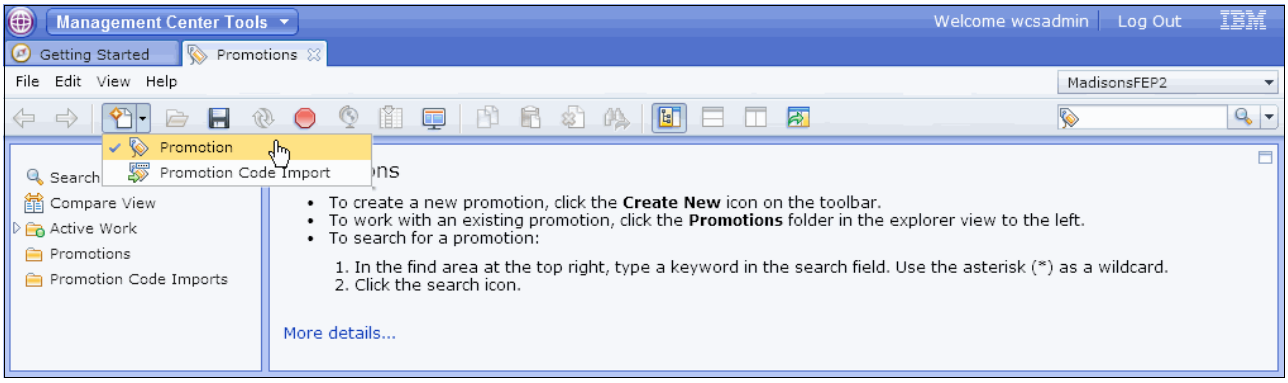


Figure 3-68 Create a new promotion

4. Select **Order promotions** from the Promotion Type Selector.

5. For Name, choose **Percentage off an order** and click **OK**, as shown in Figure 3-69.

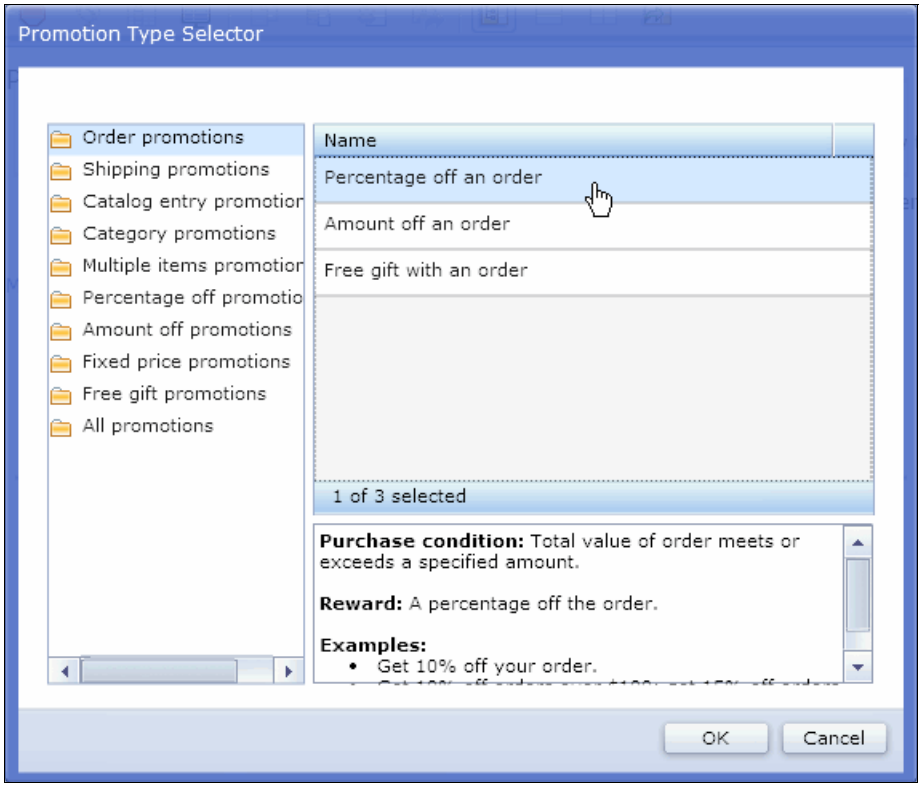


Figure 3-69 Select the promotion type

6. Enter the following values in the Promotion Properties section, as shown in Figure 3-70:
   a. For Administrative name, enter `15% off orders`.
   b. For Redemption method, select **Coupon promotion**.
   c. For Number of days until coupon expires, select **90**.
   d. For Priority, select **200**.

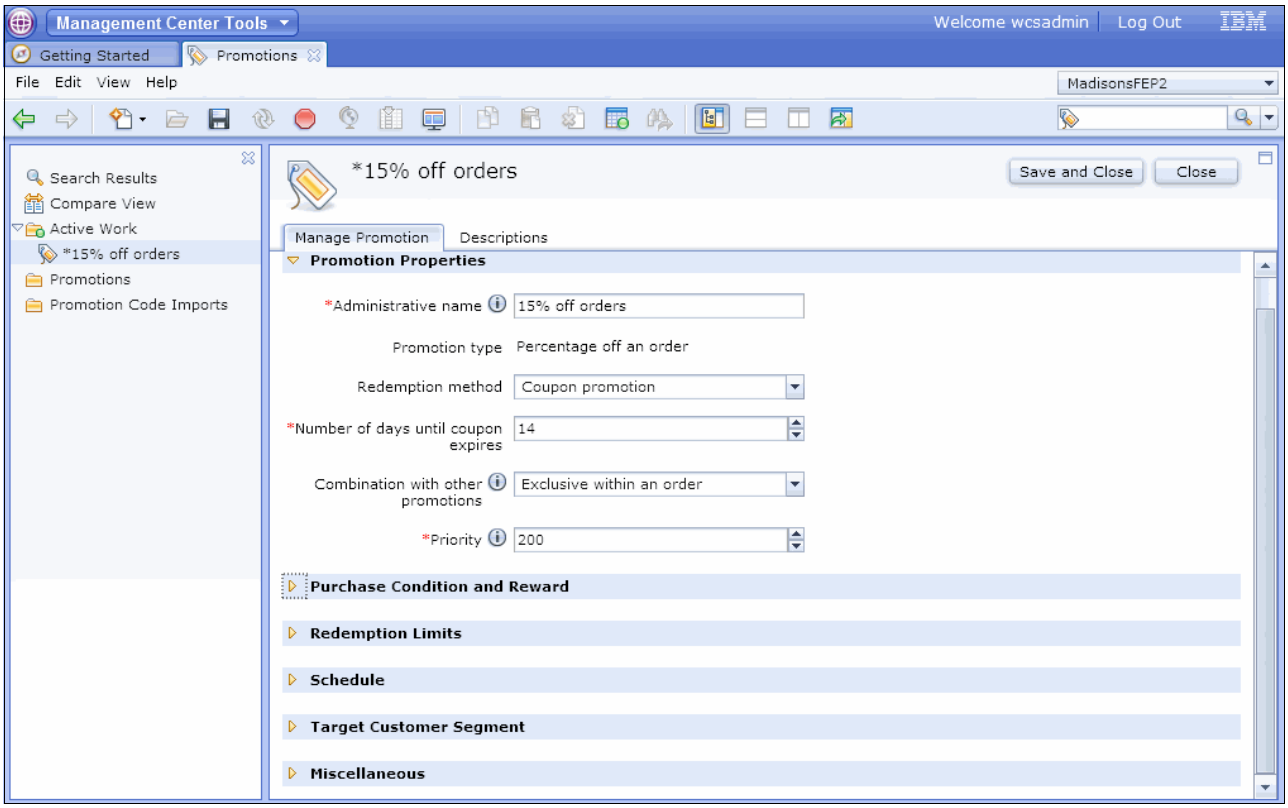


Figure 3-70 Enter the promotion properties

7. Enter the following values in the purchase condition and reward section as shown in Figure 3-71:
   a. Minimum purchase condition:
      i. For Minimum Order Purchase, enter `10.00`.
      ii. For Percentage Discount on Order (%), enter `15`.
      iii. For Maximum Discount Amount, enter `5000`.
   b. For Target payment type, select **Any payment type**.

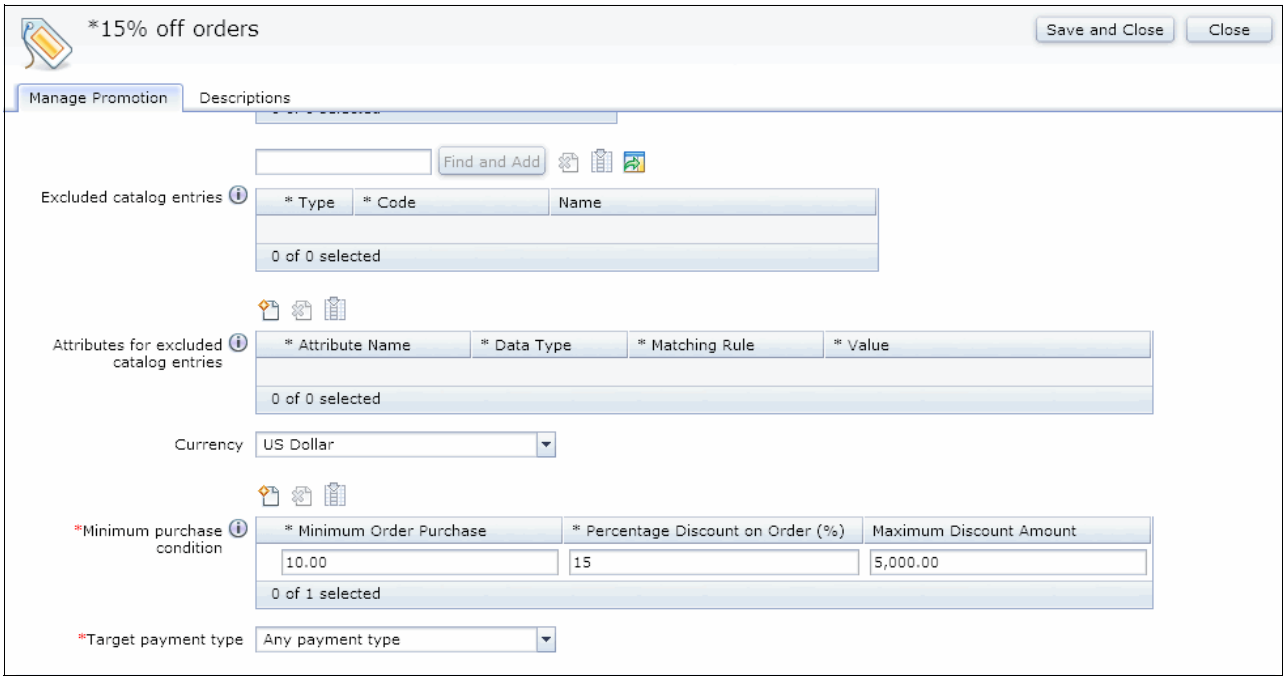


Figure 3-71 Enter the purchase conditions

8. In the **Descriptions** tab, enter the value `15% off orders` for the Customer viewable long description, as shown in Figure 3-72.

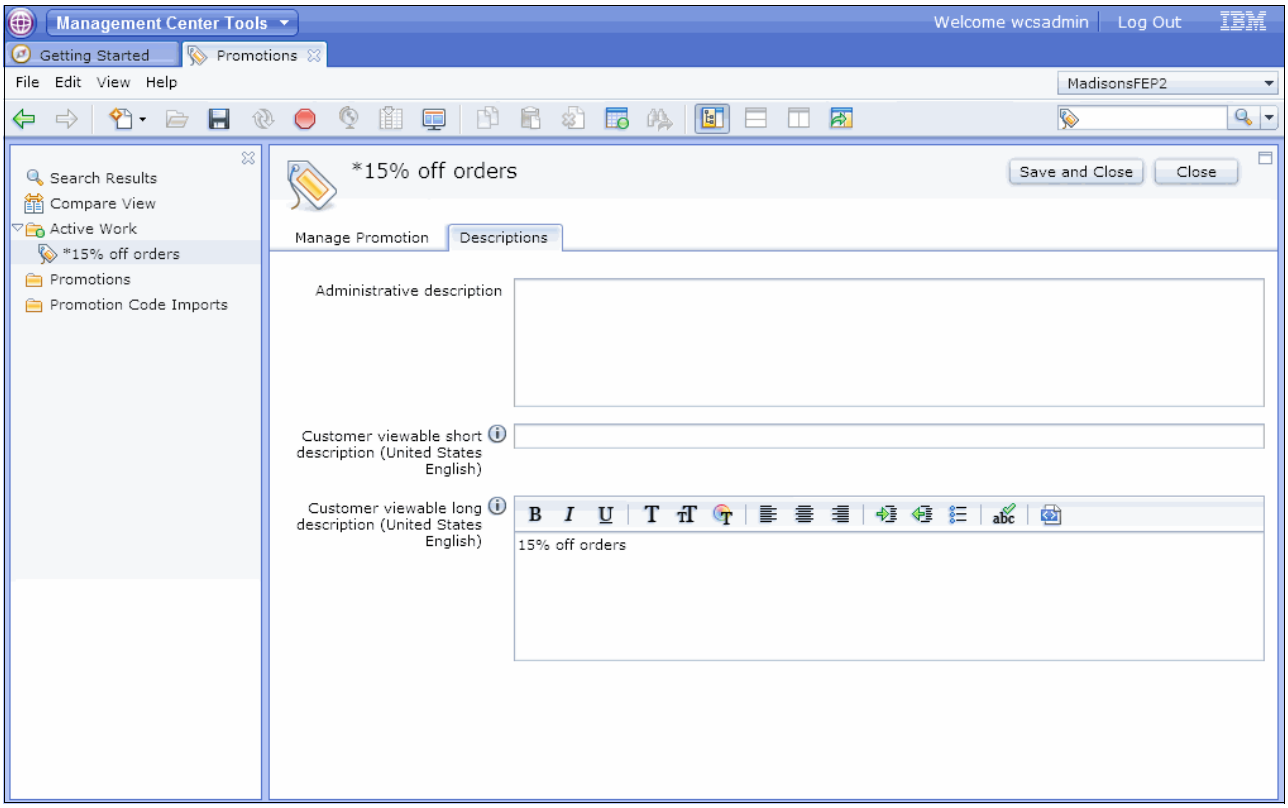


Figure 3-72 Enter the description

9. Click **Save and Close**.
10. Select **Promotions** in the explorer view.

11. Right-click the promotion from the Promotions - List view and click **Activate**, as shown in Figure 3-73.

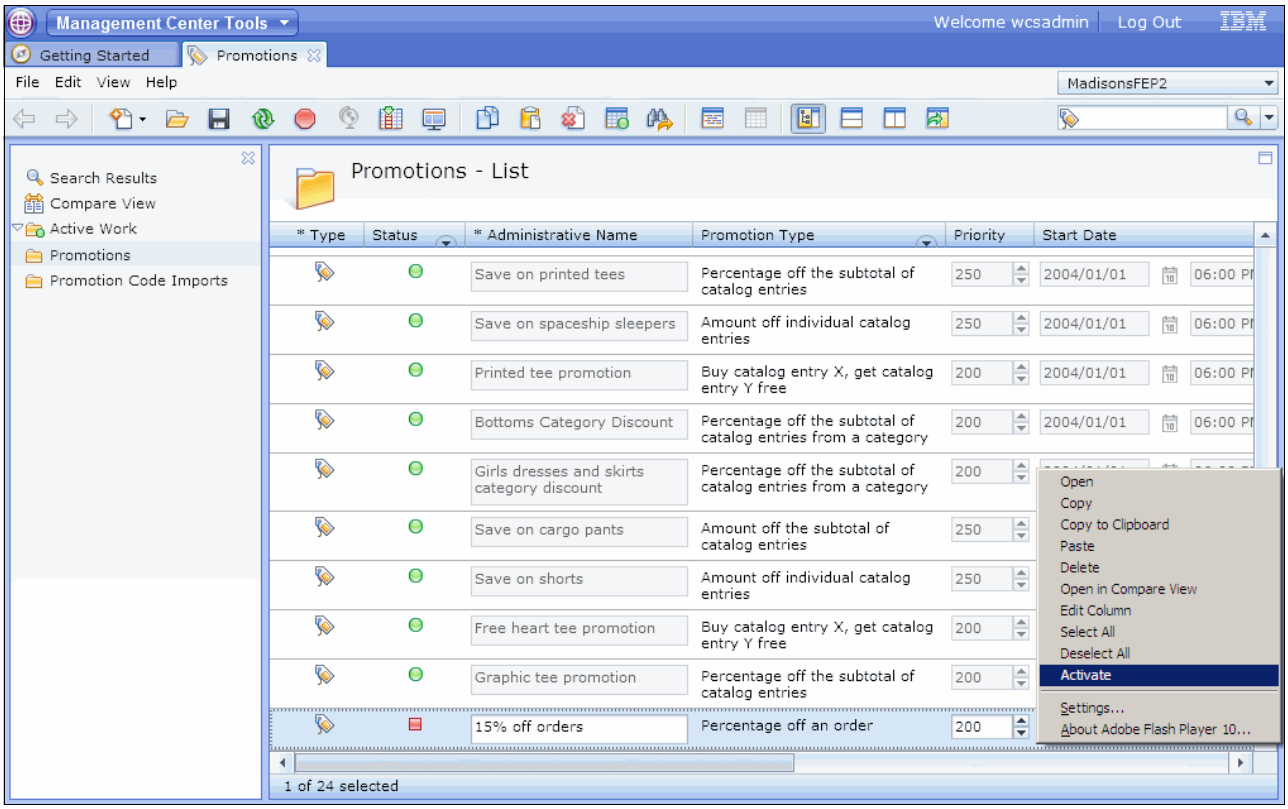


Figure 3-73 Activate the promotion

Follow these steps to create the dialog activity that triggers the issue of the coupon promotion that was created:

1. Log on to the Management Center and open the **Marketing** tool.

2. Select **Dialog Activity** from the Create New toolbar icon, as shown in Figure 3-74.

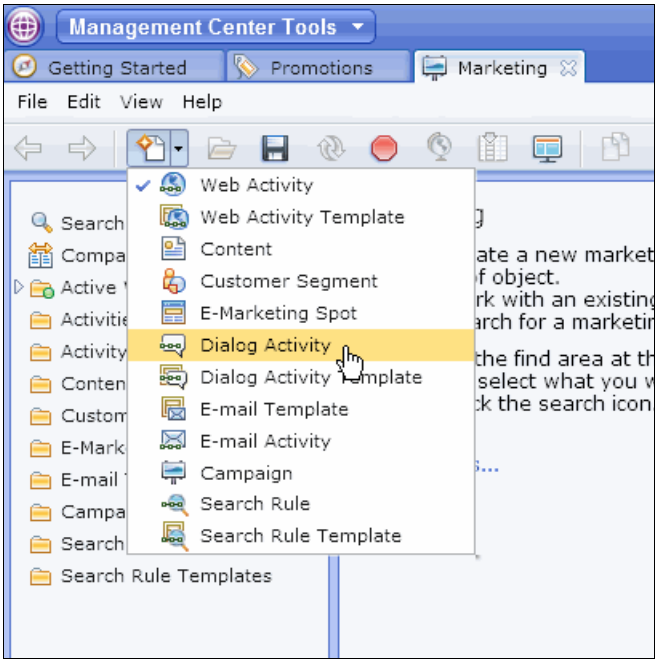


Figure 3-74 Create Dialog Activity

3. Select the **Blank Dialog Activity** template and click **OK**, as shown in Figure 3-75.

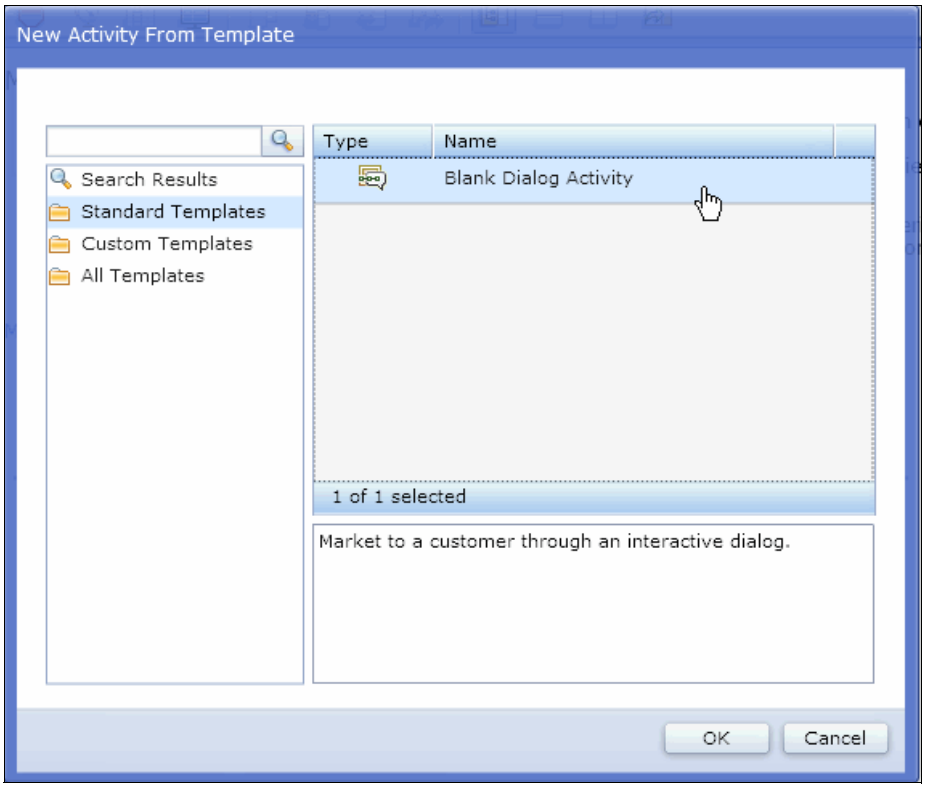


Figure 3-75 Select blank template

4. On the Dialog Activity General Properties tab, enter the name as `Apparel search coupon`, as shown in Figure 3-76.

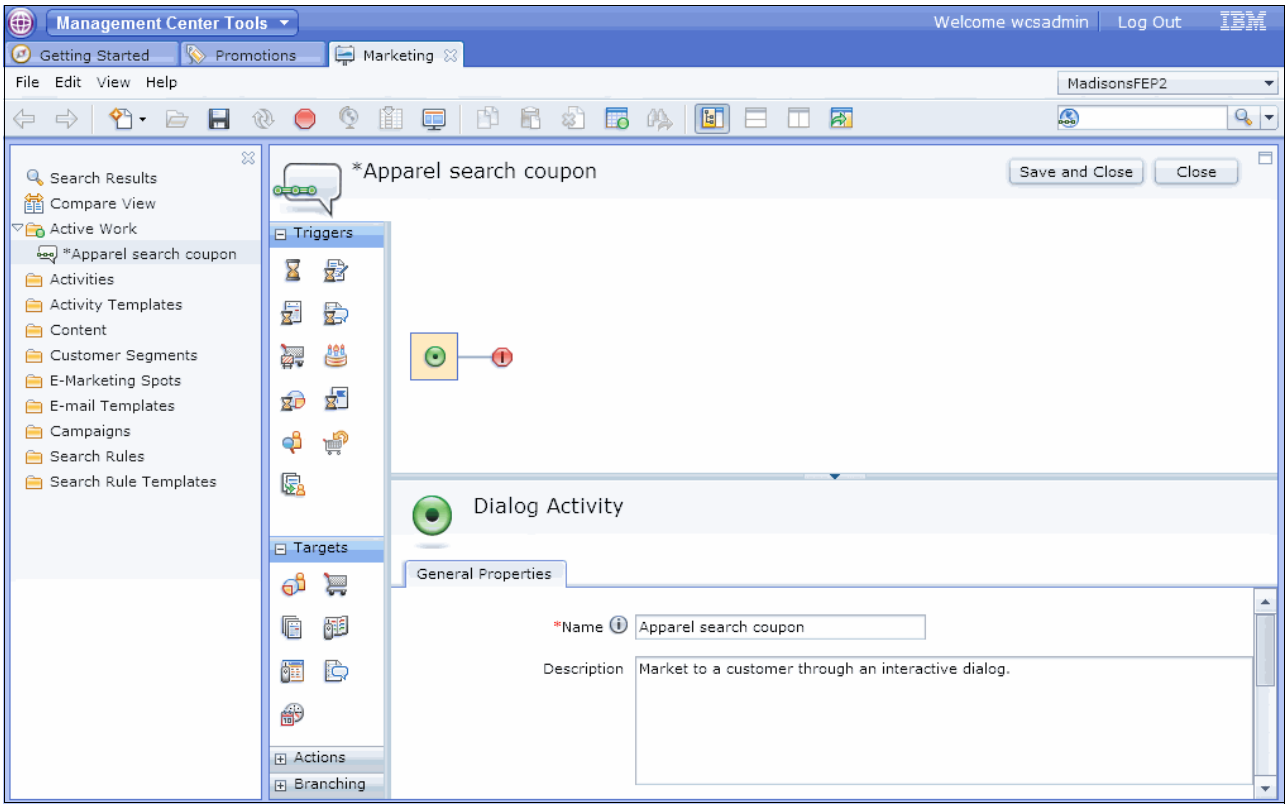


Figure 3-76 Enter the dialog properties

5. Drag and drop the **Customer Searches** trigger into the work area and click it.

6. Click the **Customer Searches** trigger to display its property panel. Enter the property values, as shown in Figure 3-77 and Figure 3-78 on page 88:

   a. For Keyword matching rule, select **Search keyword is exactly one of the following values**.

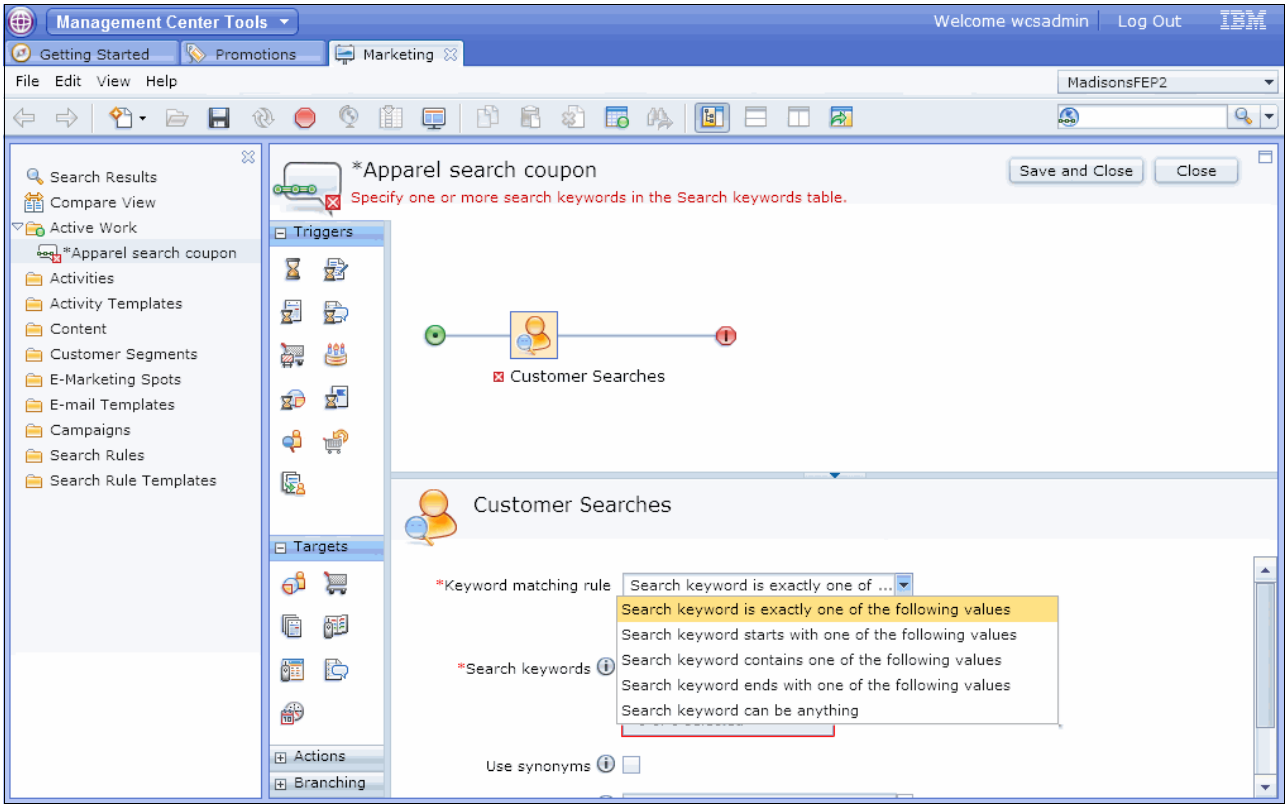


Figure 3-77 Drag and drop the trigger and select the keyword matching rule

b. For Search keywords, enter `bodysuit`.

c. Make sure that the Use synonyms option is checked. (By checking the Use synonyms check box, this activity will be triggered by the specified search keyword or any synonyms that are defined for the search keyword.)

d. For Times, select **1**.

e. For Time frame, select **At any time**.

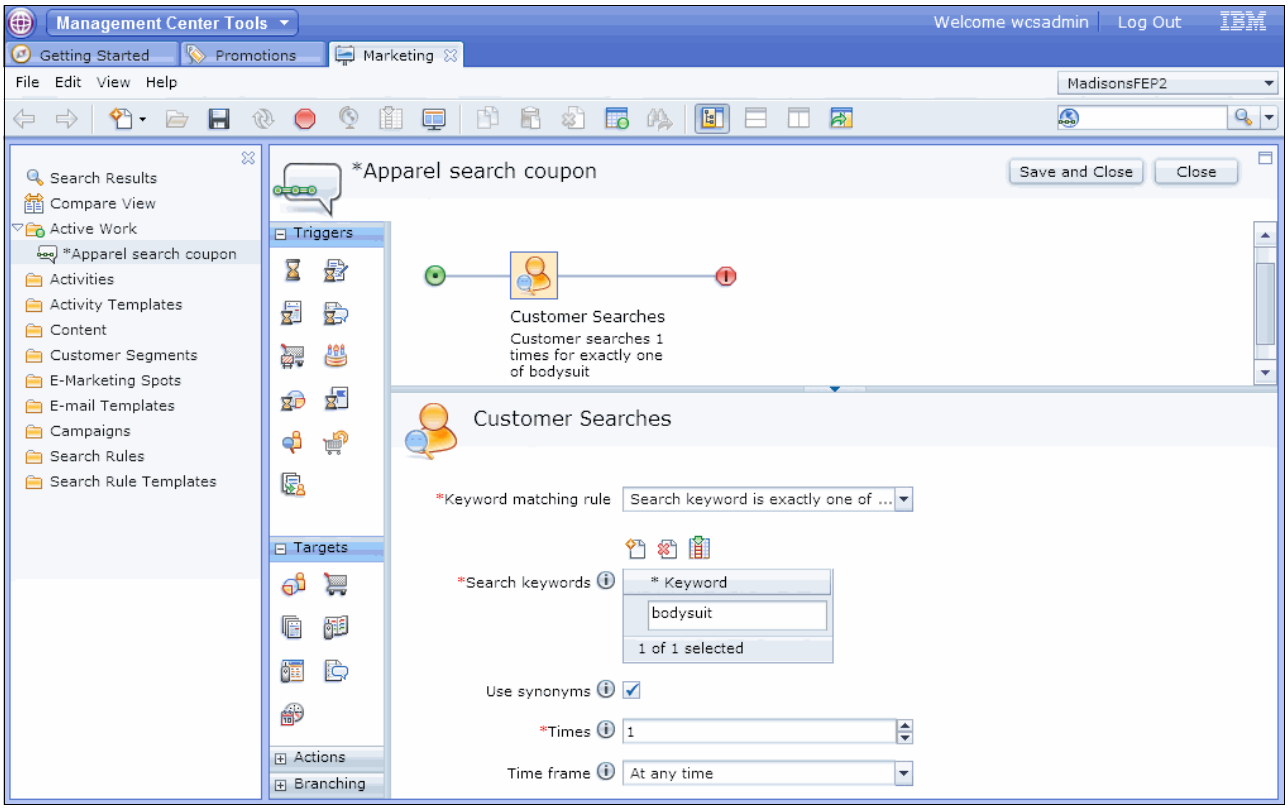


Figure 3-78 Enter the properties for the trigger

7. Drag and drop the **Issue Coupon** action into the work area after the trigger Customer Searches and click it.

8. Click the **Issue Coupon** action to display its property panel.

9. Type `15% off orders` in the text box and click **Find and Add**, as shown in Figure 3-79.

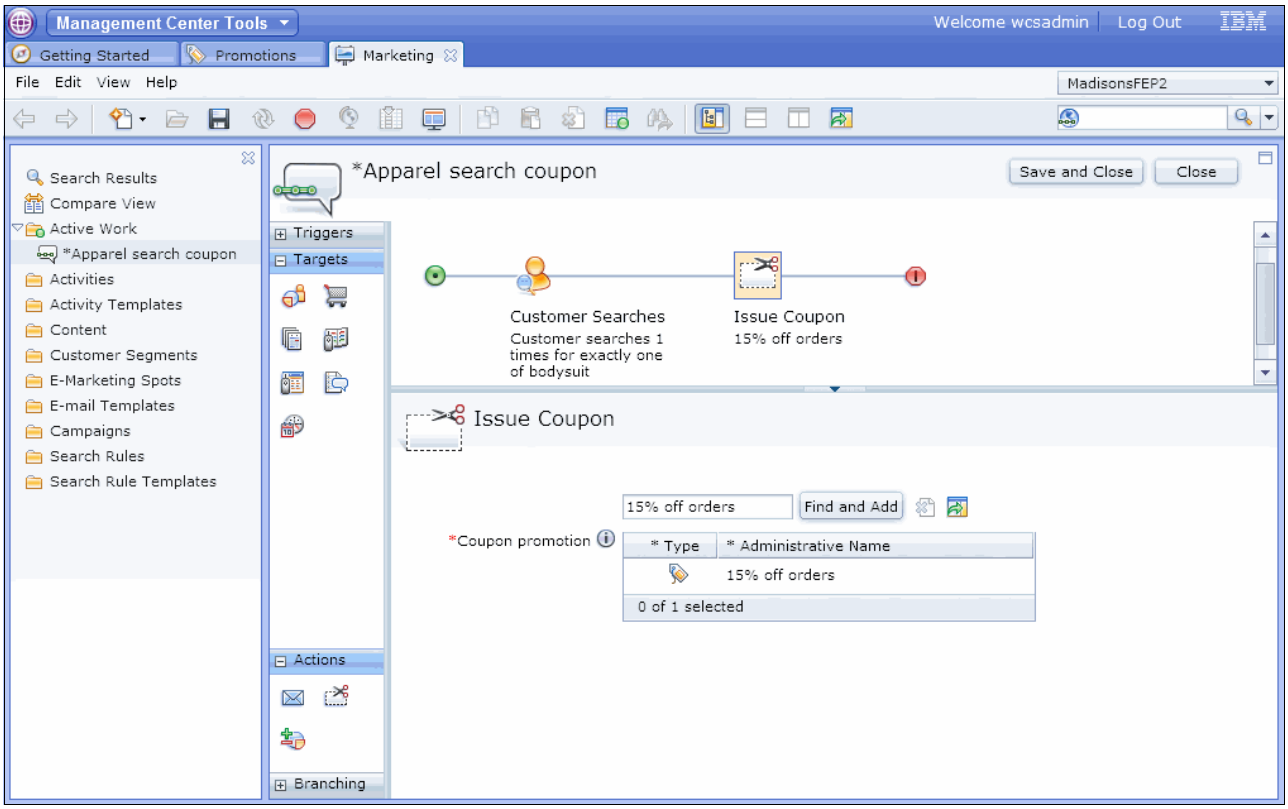


Figure 3-79 Add the Issue Coupon action

10. Click **Save and Close**.

11. Open the Activities - List, right-click the current activity that was created, and select **Activate**, as shown in Figure 3-80.

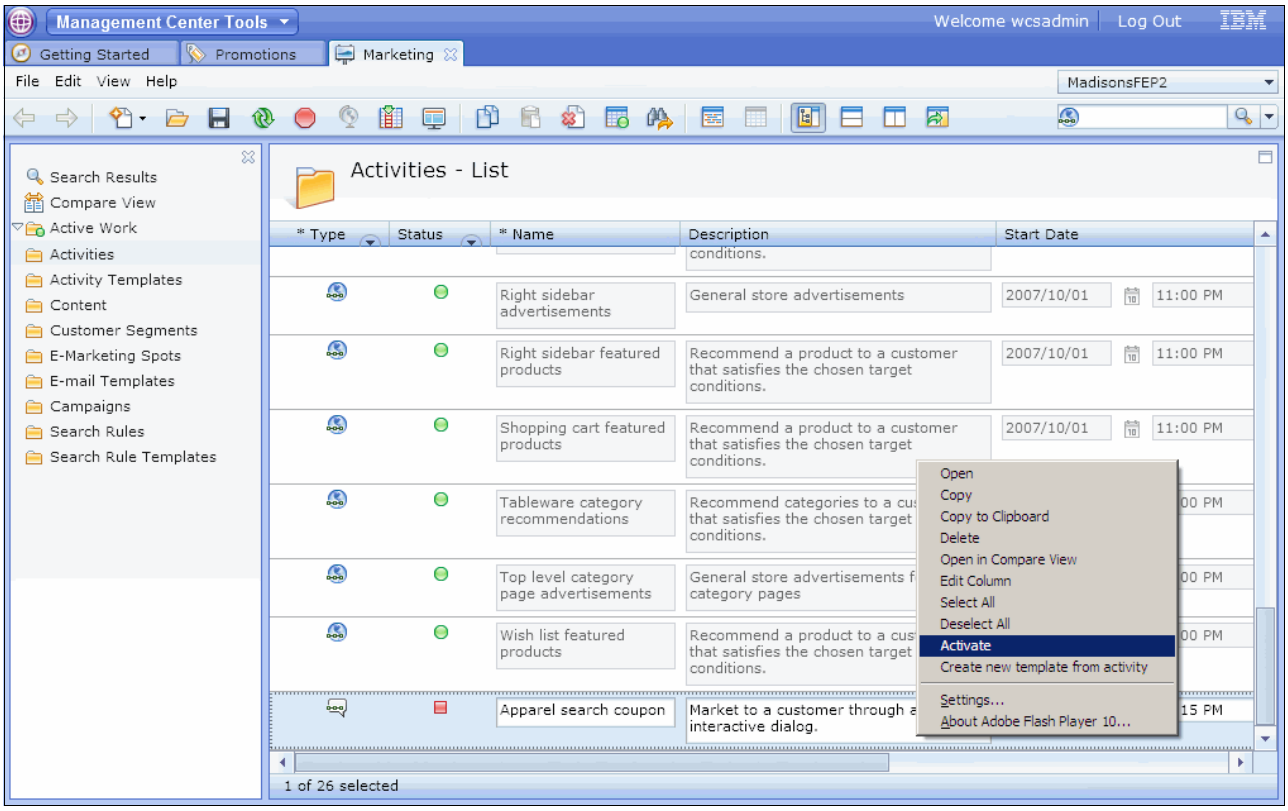


Figure 3-80 Activate the dialog activity

For the trigger Customer Searches to work, the Management Center marketing features have to be enabled and WebSphere Commerce must be configured to track the online behavior of shoppers. To enable these functions, you must follow the steps that are described in the following WebSphere Commerce Information Center link:

- Persistent sessions and personalization ID (Step 3)
- Sensor Event Listener (Step 4a - c)

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.management-center.doc/tasks/tsbenable_dup.htm

After restarting the WebSphere Commerce application or test server, log in to the storefront and search for `bodysuit`. The marketing activity is triggered and a coupon is issued to the customer.

The customer can view the coupon in the My Coupons section under the My Account page, as shown in Figure 3-81.

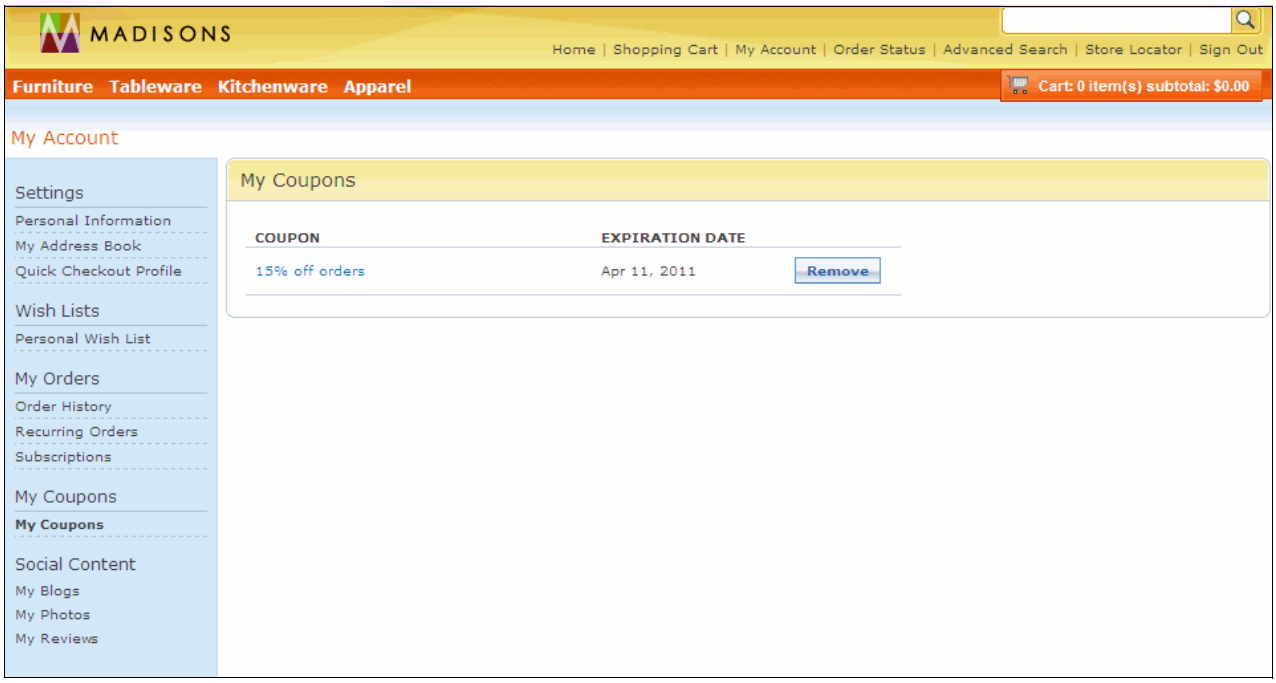


Figure 3-81 My Coupons list

When you click the coupon, you see the coupon details, as shown in Figure 3-82.

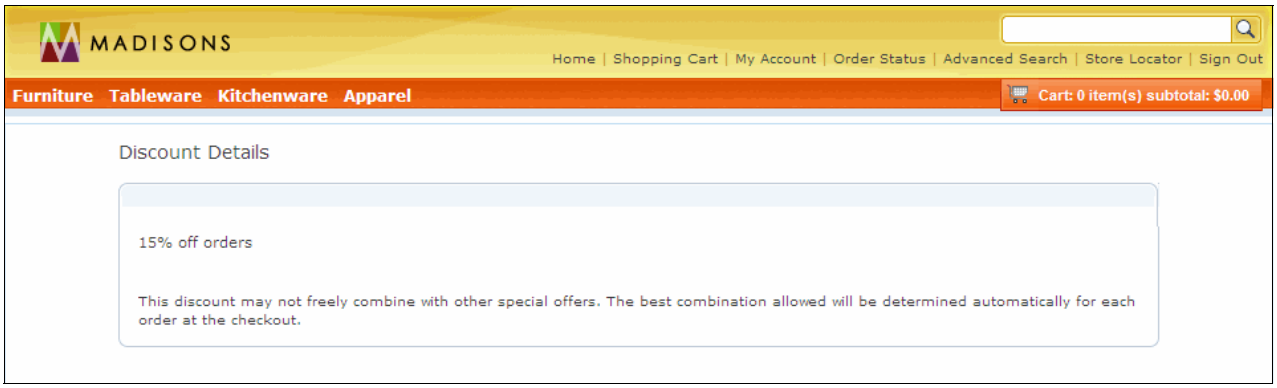


Figure 3-82 Coupon details

4

Search index life-cycle management

This section discusses the various tasks that must occur during index life-cycle management.

4.1 Full and delta index building

This section discusses full and delta index building.

4.1.1 Full versus delta index building

Making changes to a master catalog requires that the index is updated. You update the index in two steps: first run the `di-preprocess` utility and then run the `di-buildindex` utility. For more information about this updating process, refer to the WebSphere Commerce Information Center link:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdmanagesearchtop.htm

You can run these utilities to perform a delta update, or a full update. Full updates re-index the entire master catalog (you can specify which languages to re-index. The default is `all`). By default, the `di-preprocess` and `di-buildindex` utilities are configured to run a full re-index. To run a delta re-index, add the parameter `-fullbuild false` to the utility invocation.

Delta updates are tracked similarly to stagingprop; a table named TI_DELTA_CATENTRY records which catalog entries are updated. For example, if a product with `catentry_id 14401` is added to the master catalog via the Management Center, a record is inserted into the TI_DELTA_CATENTRY table, such as the record in Table 4-1.

Table 4-1 TI_DELTA_CATENTRY delta update

MasterCatalog_Id	Catentry_Id	Action
10451	14401	'U'

The U under the Action column indicates a change or update. The actual update is not recorded here. Instead, the table is simply a list of which catentries need to be updated. Therefore, a delta update still needs to query the regular database tables (catentry, catgroup, and so on) to get the actual update during `di-preprocess`. Also, unlike the staglog table that is used in stagingprop, there is no `processed` field. After the `di-preprocess` completes, the updated entries are simply deleted from TI_DELTA_CATENTRY.

Certain changes to the master catalog require a full update. For example, assigning a category to a separate parent creates the following entry in the TI_DELTA_CATENTRY table (Table 4-2).

Table 4-2 TI_DELTA_CATENTRY full update

MasterCatalog_Id	Catentry_Id	Action
10451	-1	'F'

The F under the Action column indicates that a full update is required. Note the catentry here is -1.

For a list of which common business tasks require delta or full updates, refer to this WebSphere Commerce Information Center document:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/refs/rsdsearchindexhints.htm

When the table TI_DELTA_CATENTRY contains an F entry, it is best to perform a full re-indexing, instead of a delta re-index, because, with an F entry in the

TI_DELTA_CATENTRY table, no further U entries are recorded until the F entry is cleared. The F entry is only cleared after performing a full re-indexing. Remember, therefore, when making updates to the catalog, if the re-indexing window that is scheduled is only large enough to perform a delta update, leave the changes that cause an F update until later.

Perform full re-indexing periodically even if there is not an F requiring a full index build in the TI_DELTA_CATENTRY table. When many delta updates have been made to an index, without any full index builds, the index can start to degrade due to fragmentation, which affects performance. A full index update helps with fragmentation.

4.1.2 Automatic index synchronization

You can configure both delta and full index building to occur automatically, triggered when changes are made to the catalog. This automatic delta and full index building saves you from having to manually call `di-preprocess` and `di-buildindex` to update the indexes.

By default, only the delta updates are automatically triggered. With this function enabled, if any delta updates are added to the TI_DELTA_CATENTRY table, delta re-indexing triggers the next time that the storefront is queried.

You can toggle delta and full automatic index synchronization in the configuration file `wc-search.xml`, which is located at *`WC_ear_dir`*`/xml/confil/com.ibm.commerce.catalog-fep/wc-search.xml`.

Example 4-1 shows the section that controls automatic re-indexing.

Example 4-1 Automatic index synchronization

```
<_config:index name="CatalogEntry"

object="com.ibm.commerce.catalog.facade.server.services.search.metadata.solr.SolrC
atalogNavigationViewImpl"
      deltaUpdate="true" fullBuild="false">
   </_config:index>
```

Automatic index synchronization occurs in this sequence:

1. A business user performs an update to an existing product in the Management Center. This service request triggers a ChangeCatalogEntry event to be issued.
2. A pre-configured ChangeCatalogEntry event consumer then analyzes the request and determines which type of re-indexing is required.
3. After the re-indexing type is identified, a separate index update event, ChangeCatalogNavigationView, is generated for that product and queued for the next index synchronization.
4. When the business user previews the change from the storefront, the first search request triggers an index synchronization event, which causes the search index to be re-indexed through a separate background process.

You can read about this process at the following WebSphere Commerce Information Center page:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdsearchcontentstructureindex.htm

Events (for example, changing the price of a catentry) are defined as either Delta or Full through the use of event analyzers in the configuration file `wc-event.xml`, which is located under `<WC-ear>/xml/config/com.ibm.commerce.catalog-fep./wc-event.xml`.

Example 4-2 shows several examples of event analyzers in `wc-event.xml`.

Example 4-2 Several change analyzers in the wc-event.xml configuration file

```
<!--
           CatalogEntryEventForGeneralChangeAnalyzer

           This class is the implementation of the event analyzer for detecting general
           changes in the catalog entry, such as manufacturer name, SKU, display
           sequence, etc. If so, a delta update can be performed.
         -->
        <_config:param
name="com.ibm.commerce.catalog.facade.server.event.analyzer.CatalogEntryEventForGeneralChangeAna
lyzer" />

        <!--


           CatalogEntryEventForAttachmentChangeAnalyzer

           This class is the implementation of the event analyzer for detecting changes
           in the catalog attachment. If so, a full update can be performed.
         -->
        <_config:param
name="com.ibm.commerce.catalog.facade.server.event.analyzer.CatalogEntryEventForAttachmentChange
Analyzer" />
```

4.1.3 Index building configuration

Index building is straightforward when there is only one search server and it is located on the same box as the commerce server. However, when working with multiple search servers that are located on remote machines, it is important to understand the configuration files and tables on both the commerce server and each of the search servers.

Depending on your search server configuration, you might have a search server that is dedicated to only building indexes, and not to serving queries. You can configure this dedicated search index server to push newly built indexes out to the dedicated search servers.

One possible configuration is to make the search indexer part of your staging environment. Configure this server to query the staging database during index building, rather than the production database. This setup is best for databases that are under a heavy, constant load and for servers that experience consistently high traffic.

Figure 4-1 on page 97 shows this configuration.

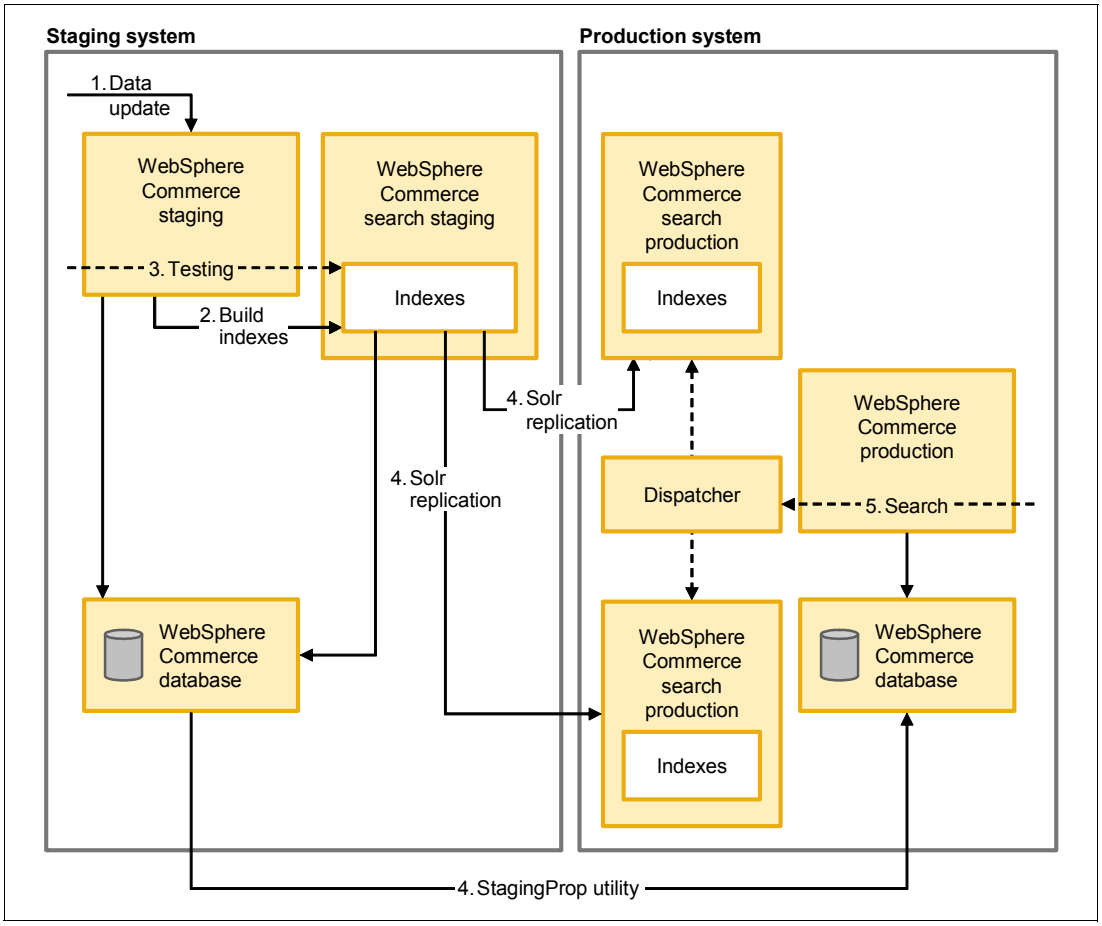


Figure 4-1 Typical WebSphere Commerce search deployment for a large index size

Regardless of the configuration, building indexes always starts with `di-preprocess` and `di-buildindex`. These indexing commands are initiated on the *commerce server*, not the search server. For these commands to work, WebSphere Commerce needs to know which server is the actual indexing machine, which is set up by running `SetupSearchIndex.sh`.

For advanced configurations in which the search server is remote from the commerce server, `SetupSearchIndex.sh` is called at least twice for each master catalog:

- On the commerce server
- On the search indexing machine
- And on each of the search machines that holds the catalog. (You can actually skip calling `SetupSearchIndex.sh` on these machines, because you can simply copy the Solr home directory from the search indexing machine.)

On the commerce server, `SetupSearchIndex.sh` needs to be called with the parameter `action` set to `configWCforSolr`. This action configures several sections in the `wc-search.xml`, and populates the table SRCHCONF. WebSphere Commerce uses the SRCHCONF table to look up which Solr server is responsible for building which index. For example, Table 4-3 on page 98 shows a section of the table for one catalog.

Table 4-3 IndexScope and Config column from SRCHCONF

IndexScope	Config
10001	IndexScopeTag=0,SearchServerPort=3737,SearchServerName=linux-530s, PreProcessConfigDirectory=/opt/IBM/WebSphere/CommerceServer70/instances/demo/search/pre-processConfig/MC_10001/DB2

The IndexScope is for master catalog 10001. The Config column contains several pieces of information:

- The `IndexScopeTag` is used at the end of temporary tables, to identify which temporary tables belong to which index. The tables for the index, 10001, have 0 appended to them, for example, TI_CATENTRY_0.
- `SearchServerPort` and `SearchServerName` are used to point the **`di-preprocess`** and **`di-buildindex`** commands to the proper Java virtual machine (JVM).
- `PreProcessConfigDirector` contains several files that were constructed during the `setupSearchIndex.bat` to be used during `di-preprocess`.

If the indexing machine is moved, the SRCHCONF table needs to be updated, to point again to the proper machine.

On the Solr machine, the **`SetupSearchIndex.sh`** (action - configSolrCores) command simply sets up the proper directory structure for the index. For this reason, after you have run this command on one machine, you can copy the directory structure to the other Solr servers, rather than running the command again.

4.1.4 Replication

If you use a dedicated indexing machine, after the dedicated indexer completes building an index, it needs to propagate this index to the dedicated search machines. You can obtain the steps to configure this replication in the following WebSphere Commerce Information Center document:

`http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdreplicatesearch.htm`

With replication configured from the indexing machine to the dedicated search servers, you need to configure the `wc-search.xml` file for WebSphere Commerce to send queries to the proper machine. The `wc-search.xml` file contains the mapping instructions for WebSphere Commerce to know where to send each query. How you configure the `wc-search.xml` file really depends on your server configuration and index distribution. In this book, we outline a scenario in which two search servers each hold separate cores (see 6.17.1, “Setting up language cores on separate Solr servers” on page 173). Another perfectly valid scenario is to have every index on every search server and to set up a load balancer to equally distribute the search requests.

It is possible to cluster Solr JVMs. This search cluster is separate from the commerce cluster, because it has a separate profile and a separate enterprise archive (EAR) file. You can obtain the instructions to set up the clustering for Solr servers in the WebSphere Commerce Information Center:

`http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdfederatesearch.htm`

Important: Even with clustered Solr servers, you still must set up all of the machines in the cluster for replication. A WebSphere Application Server cluster only manages the Solr application, not the search index data files. These data files reside locally on each node and are not shared across the cluster members.

5

Troubleshooting

This chapter provides an outline of how to approach troubleshooting search-related issues, on the Solr server and on the WebSphere Commerce server. The chapter concludes with issues, which we encountered during the Solr deployment and maintenance, and the recommended solutions.

5.1 Starting the Solr server

After you deploy and start the search server, check the `SystemOut.log` to verify that the search server started properly. In standard and advanced deployment mode, the search server has its own profile, which is separate from the commerce profile, and so has a separate set of logs. These logs are located on the Solr server machine, in the directory:

`/opt/IBM/WebSphere/AppServer/profiles/`*`search_profile`*`/logs/`*`solrServer`*`/`

In the `SystemOut.log`, search for this message:

```
WSVR0221I: Application started: solr
```

If the Solr server was unable to start, this message is printed in the log:

```
SolrDispatchF E org.apache.solr.servlet.SolrDispatchFilter init Could not start
SOLR.
```

The message is followed by error details, for example:

```
SolrDispatchF E org.apache.solr.servlet.SolrDispatchFilter init Could not start
SOLR. Check solr/home property
```

In Example 5-1, the `/solr/home` directory was not set in the Java Naming and Directory Interface (JNDI).

Example 5-1 SystemOut.log error details

```
[3/30/11 14:56:59:981 EDT] 0000001c SolrResourceL I org.apache.solr.core.SolrResourceLoader
locateSolrHome No /solr/home in JNDI
[3/30/11 14:56:59:982 EDT] 0000001c SolrResourceL I org.apache.solr.core.SolrResourceLoader
locateSolrHome solr home defaulted to 'solr/' (could not find system property or JNDI)
[3/30/11 14:56:59:982 EDT] 0000001c SolrResourceL I org.apache.solr.core.SolrResourceLoader
<init> Solr home set to 'solr/'
[3/30/11 14:57:00:065 EDT] 0000001c SolrDispatchF I org.apache.solr.servlet.SolrDispatchFilter
init SolrDispatchFilter.init()
[3/30/11 14:57:00:066 EDT] 0000001c SolrResourceL I org.apache.solr.core.SolrResourceLoader
locateSolrHome No /solr/home in JNDI
[3/30/11 14:57:00:066 EDT] 0000001c SolrResourceL I org.apache.solr.core.SolrResourceLoader
locateSolrHome solr home defaulted to 'solr/' (could not find system property or JNDI)
[3/30/11 14:57:00:068 EDT] 0000001c CoreContainer I
org.apache.solr.core.CoreContainer$Initializer initialize looking for solr.xml:
/opt/IBM/WebSphere/AppServer/profiles/demo_solr/solr/solr.xml
[3/30/11 14:57:00:069 EDT] 0000001c SolrResourceL I org.apache.solr.core.SolrResourceLoader
<init> Solr home set to 'solr/'
[3/30/11 14:57:00:072 EDT] 0000001c SolrDispatchF E org.apache.solr.servlet.SolrDispatchFilter
init Could not start SOLR. Check solr/home property
                              java.lang.RuntimeException: Can't find resource
'solrconfig.xml' in classpath or 'solr/conf/',
cwd=/opt/IBM/WebSphere/AppServer/profiles/demo_solr
```

Adding the `/solr/home` location to the JNDI resolved this error. After the Solr server has been started, ensure that you can access the Solr server. In a browser, enter the URL:

`http://`*`host_name`*`:3737/solr/Default/select?q=*%3A*`

In this URL, *host_name* is the name of the search server. Port 3737 is the default port that is opened during deployment.

You then see a response, similar to Example 5-2.

Example 5-2 Solr server response

```
<response>
.
<lst name="responseHeader">
<int name="status">0</int>
<int name="QTime">140</int>
.
<lst name="params">
<str name="q">*:*</str>
</lst>
</lst>
<result name="response" numFound="0" start="0"/>
</response>
```

If you see an error instead, try to narrow down where the problem is occurring. Try to access the web server in front of the Solr server directly. Check the `SystemOut.log` to see if solr is receiving the query. The generic solr query in Example 5-2 results in an entry that is similar to the following log entry in the `SystemOut.log`:

```
org.apache.solr.core.SolrCore execute [Default] webapp=/solr path=/select
params={q=*:*} hits=0 status=0 QTime=0
```

5.2 Querying cores

You can query indexes directly from the Solr server by using only a browser, or you can run queries through the Commerce storefront. This section shows you both methods.

5.2.1 Querying cores directly on the Solr server

After indexing a master catalog, you can use a web browser to query the search server directly, instead of searching through the storefront. When troubleshooting indexes, first try the search queries directly on the search server. This step can help you determine if the issue is with the index data, or on the Commerce storefront side.

Queries on indexes use the following syntax:

`http://host_name:3737/solr/`*`core_name`*`/select?q=query`

The *core_name* is defined as a default by Commerce as:

`MC_`*`MasterCatalogID`*`_CatalogEntry_`*`locale`*

Indexes are separated by `Master Catalog` and `locale`. A master catalog with three languages has three separate cores for structured content.

The simplest query is *`field name: field value`*. For example, to search for a specific catalog Entry ID, for example, `100102`, enter this command:

```
http://host_name:3737/solr/MC_10001_CatalogEntry_en_US/select?q=catentry_id:100102
```

Make sure to use the field names as defined for the index, and not the associated database Column name. For example, in the table CATENTRY, there is a column called `PARTNUMBER`.

However, using the field name PARTNUMBER (that is, q=*PARTNUMBER:12102)* results in a "page not found" error.

Each core has a wc-data-config.xml configuration file that contains the field column mappings. You can locate this file in:

solr/home/MC_*MasterCatalogID*/*language*/CatalogEntry/conf/wc-data-config.xml

In this file, you can see that the database column PARTNUMBER maps to the index field partNumber_ntk:

```
<field column="PARTNUMBER" name="partNumber_ntk" />
```

The field names that are used in query searches are also *case-sensitive*. Using an uppercase P, for partNumber_ntk, for example, q=**P**artNumber_ntk:12102, results in "a page not found" error.

Look at the query result set. To search in master catalog 10001 in English for catentries with Jeans in the name, use this query:

```
http://host_name:3737/solr/MC_10001_CatalogEntry_en_US/select?q=name:Jeans
```

The query results set returns every full document where the field name contains Jeans. At the start of the result set, response statistics, which are useful for checking the overall performance of the query, are returned. See Example 5-3.

Example 5-3 Response header

```
<?xml version="1.0" encoding="UTF-8" ?>
- <response>
- <lst name="responseHeader">
<int name="status">0</int>
<int name="QTime">0</int>
- <lst name="params">
<str name="q">name:Jeans</str>
</lst>
</lst>
- <result name="response" numFound="21" start="0">
```

The response statistics are followed by the resulting documents. Example 5-4 displays one full document that is returned.

Example 5-4 Solr document

```
- <doc>
<int name="buyable">1</int>
+ <arr name="catalog_id">
<long>10451</long>
</arr>
<long name="catentry_id">12850</long>
<str name="catenttype_id_ntk_cs">ProductBean</str>
<str name="fullImage">images/catalog/apparel/apparel_160x160/IMG_0048_e.jpg</str>
<long name="member_id">7000000000000000901</long>
<str name="mfName">MapleWear</str>
<str name="mfName_ntk">MapleWear</str>
<str name="mfName_ntk_cs">MapleWear</str>
<str name="name">Eagle jeans</str>
- <arr name="parentCatgroup_id_facet">
<str>10451_10432</str>
```

```
</arr>
- <arr name="parentCatgroup_id_search">
<str>10451_10432</str>
<str>10451_10429</str>
</arr>
<str name="partNumber_ntk">MW-0048</str>
<float name="price_USD">19.99</float>
<int name="published">1</int>
<str name="shortDescription">Comfy denim jeans. delta</str>
<int name="storeent_id">11051</int>
<str name="thumbnail">images/catalog/apparel/apparel_70x70/IMG_0048_e.jpg</str>
</doc>
```

The URL can accept additional parameters to modify or simplify what is returned. For example, adding the `fl` parameter allows you to limit which fields are returned:

```
http://localhost/solr/MC_10001_CatalogEntry_en_US/select?q=name:Jeans&fl=name,cate
ntry_id
```

Adding the `fl` parameter results in Example 5-5.

Example 5-5 Simplified result set

```
  - <doc>
  <long name="catentry_id">12850</long>
  <str name="name">Eagle jeans</str>
  </doc>
  - <doc>
  <long name="catentry_id">12970</long>
  <str name="name">Eagle jeans</str>
  </doc>
  - <doc>
  <long name="catentry_id">12971</long>
  <str name="name">Eagle jeans</str>
  </doc>
  ...
...
```

For further parameters, refer to Solr-specific documentation.

There are also search graphical user interfaces (GUIs) available for querying indexes, such as the GUI, Luke. Luke does not query the Solr server, instead it reads directly from the Lucene indexes. It completely bypasses Solr.

You can download Luke from this website:

http://code.google.com/p/luke/downloads/list

When opening Luke, you need to specify the index directory, as shown in Figure 5-1 on page 106.

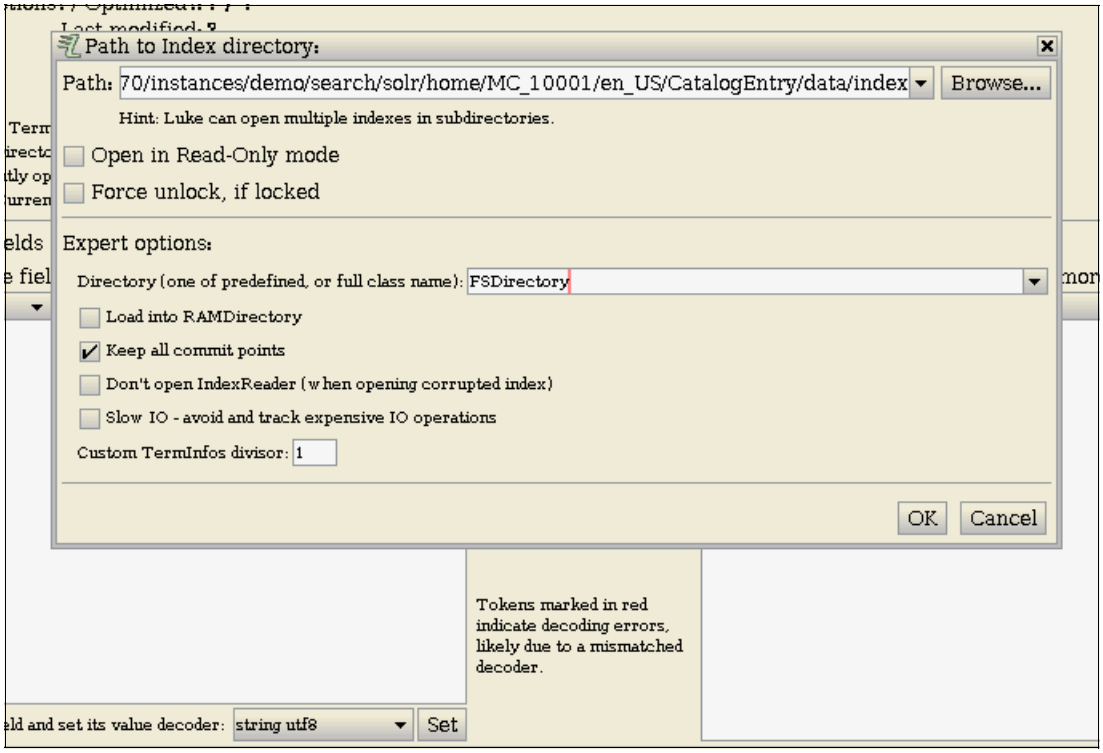


Figure 5-1 Luke index selection

You can browse documents individually, as shown in Figure 5-2 on page 107.

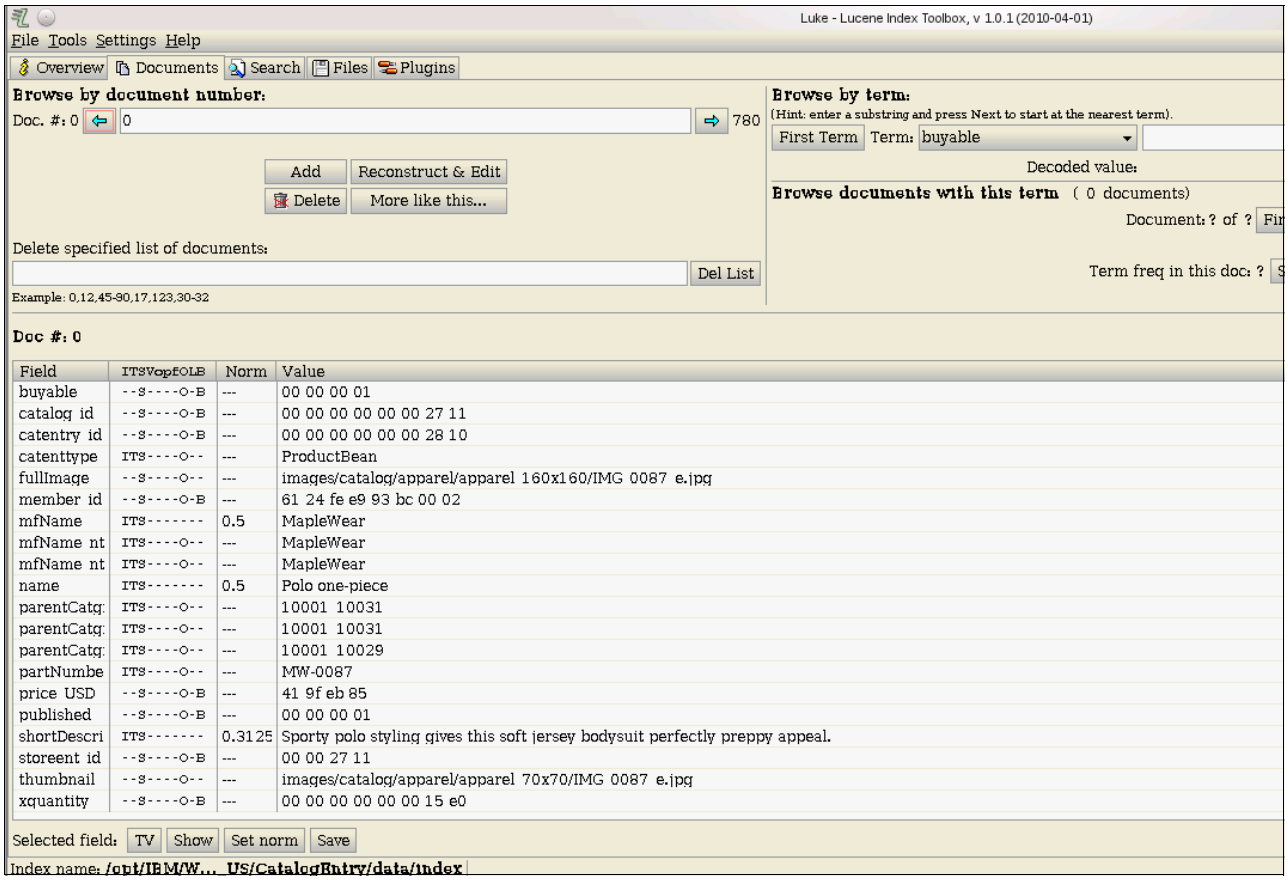


Figure 5-2 Browsing an individual document

Also, you can run queries against the index using the Search tab, as shown in Figure 5-3 on page 108.

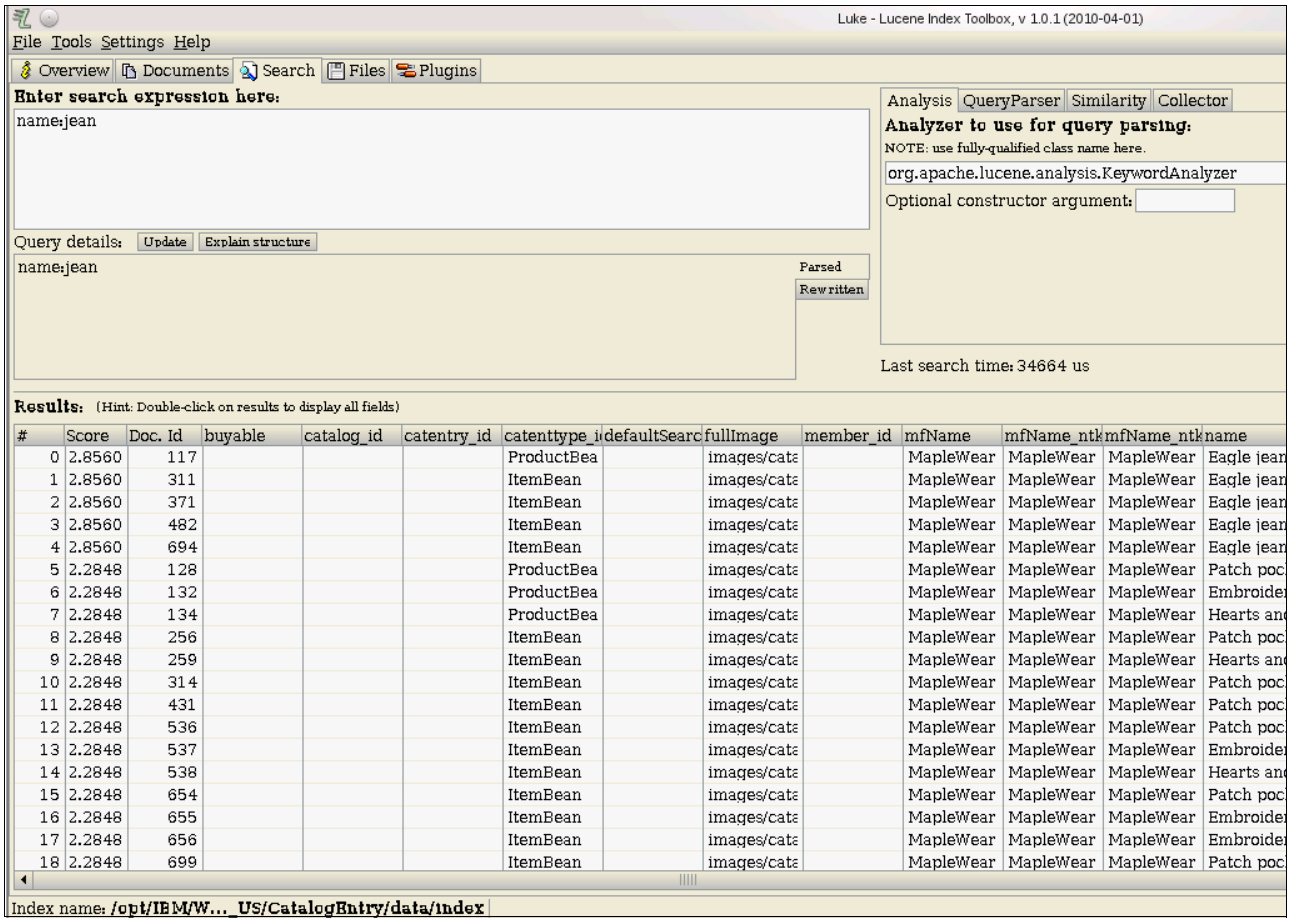


Figure 5-3 Running a query

Note that the queries are constructed differently in Luke than they are in Solr. For example, using the earlier example of searching for jeans, in a Solr browser, you use the following Luke query:

```
http://host_name:3737/solr/MC_10001_CatalogEntry_en_US/select?q=name:jeans
```

Using the `name:jeans` query in the search field produces 0 results. Oddly enough, using `name:jean` produces the correct results. You can see the same result, for example, using `coffee`. Using `name:coffee` returns zero results, but using `name:coffe` returns the correct results. Luke seems to be configured, by default, to not return exact matches using this syntax. Adding a tilde (~) after the complete search term returns the proper results, for example, `name:coffee~`.

5.2.2 Querying cores from the storefront

If you have determined that the problem is not with the indexes on the Solr server (for example, the query successfully returns the expected result set), run the query next from the storefront.

Queries on the storefront run differently than they run directly on the Solr server. The predefined, ready-to-use queries are configured in the Search Profiles option, which is located in the `wc-search.xml` configuration file in the following directory:

WC_ear_dir`/xml/config/com.ibm.commerce.catalog-fep/wc-search.xml`

Example 5-6 depicts a search profile that was taken from the predefined `wc-search.xml`.

Example 5-6 Search profile

```
<_config:profile extends="IBM_findCatalogEntryByName"
name="IBM_findCatalogEntryByNameAndShortDescription">
        <_config:query inherits="true">
            <_config:field name="defaultSearch"/>
            <_config:field name="shortDescription"/>
        </_config:query>
        <_config:highlight inherits="true">
            <_config:field name="shortDescription"/>
        </_config:highlight>
    </_config:profile>
```

This profile, which is called `IBM_findCatalogEntryByNameAndShortDescription`, extends the `IBM_findCatalogEntryByName` profile. The `IBM_findCatalogEntryByNameAndShortDescription profile` inherits all the config parameters from `IBM_findCatalogEntryByName` and all the query parameters of that profile (in this case, the `name` field).

If a search for `Jeans` on the storefront is executed using this profile, `Jeans` is searched for against the fields: `name`, `shortDescription`, and `defaultSearch`. See the `SystemOut.log`, which is similar to Example 5-7, on the Solr server.

Example 5-7 Solr server query

```
000001d2 SolrCore      I org.apache.solr.core.SolrCore execute [MC_10451_CatalogEntry_en_US]
webapp=null path=/select
params={q=%2B%28name%3A%28Jeans%29+defaultSearch%3A%28Jeans%29+shortDescription%3A%28Jeans%29%29
&fq=catalog_id%3A10451&fq=storeent_id%3A%2811051%29&fq=published%3A1&fq=-catenttype_id_ntk_cs%3A
ItemBean&start=0&rows=12&timeAllowed=3000&hl.fl=name&hl.fl=shortDescription&hl=true&hl.simple.pr
e=%3Cstrong%3E%3Cspan+class%3Dfont2%3E&hl.simple.post=%3C%2Fspan%3E%3C%2Fstrong%3E&hl.requireFie
ldMatch=true&facet.field=mfName_ntk_cs&facet.field=parentCatgroup_id_facet&facet=true&facet.quer
y=price_USD%3A%28%7B*+100%7D+100%29&facet.query=price_USD%3A%28%7B100+200%7D+200%29&facet.query=
price_USD%3A%28%7B200+300%7D+300%29&facet.query=price_USD%3A%28%7B300+400%7D+400%29&facet.query=
price_USD%3A%28%7B400+500%7D+500%29&facet.query=price_USD%3A%28%7B500+*%7D%29&facet.sort=count&f
acet.mincount=1&facet.limit=10&spellcheck.count=5&spellcheck=true&spellcheck.collate=false&debug
Query=false}
```

The query that is shown in Example 5-7 is much more complex than the simple searches that we performed directly against the Solr server in the earlier examples. When debugging why a particular query does not produce the expected results on the storefront, do not forget to consider this added complexity.

For example, in the query in Example 5-7, multiple search parameters of type `fq` are added. An `fq` is a filter query, which imposes additional constraints against the result set. The simplest filter query in Example 5-7 is `fq=published`. This filter removes any catentries that are not published from appearing in the final search results.

The following WebSphere Commerce Information Center document gives more detailed information about the search interface that Commerce uses to construct the queries to send to the Solr server:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdsearchinterface.htm

Also, on the Commerce server, you can enable tracing for the search. See Example 5-8 on page 110.

Example 5-8 Search trace string

```
com.ibm.commerce.search.*=all: com.ibm.commerce.foundation.*=all
```

This trace prints to the `trace.log` file in your Commerce server, in the directory:

`<App server install dir>/profiles/<profile_name>/logs/server1`

Example 5-9 shows the fully constructed search query for `jeans` on the Commerce server, before it is sent to Solr. When looking through the trace, search for **`Final Solr query expression`**.

Example 5-9 Search tracing on the Commerce side

```
000001d7 solr          1
com.ibm.commerce.foundation.internal.server.services.search.processor.solr.SolrSea
rchExpressionProcessor getEntityObjects Final Solr query expression:
q=+(name:(jeans) defaultSearch:(jeans)
shortDescription:(jeans))&fq=catalog_id:10451&fq=storeent_id:(11051)&fq=published:
1&fq=-catenttype_id_ntk_cs:ItemBean&start=0&rows=12&timeAllowed=3000&hl.fl=name&hl
.fl=shortDescription&hl=true&hl.simple.pre=<strong><span
class=font2>&hl.simple.post=</span></strong>&hl.requireFieldMatch=true&facet.field
=mfName_ntk_cs&facet.field=parentCatgroup_id_facet&facet=true&facet.query=price_US
D:({* 100} 100)&facet.query=price_USD:({100 200} 200)&facet.query=price_USD:({200
300} 300)&facet.query=price_USD:({300 400} 400)&facet.query=price_USD:({400 500}
500)&facet.query=price_USD:({500
*})&facet.sort=count&facet.mincount=1&facet.limit=10&spellcheck.count=5&spellcheck
=true&spellcheck.collate=false&debugQuery=true
```

Important: Foundation tracing can affect performance. Do not enable foundation tracing on production during periods of heavy traffic, if possible.

5.3 Errors encountered during search life-cycle management

In this section, we describe several issues that you might encounter while building and deploying search indexes.

Search features not working on storefront

The following reasons explain why search features might not work on a newly published store:

- Publishing `MadisonsEnhancements.sar` onto the newly published Madisons store.
- Enabling search-based navigation.
- Indexing the master catalog.
- Browsing a sales catalog that was created after the indexes were built. Rebuild the index, which will include the sales catalog in the search index.

Category-level stock keeping units (SKUs) not shown in search results

Perform the following steps to reproduce this situation:

1. In the Management Center, under the Catalog tab, select **New** → **category level SKU**.

2. Rebuild the index.
3. On the storefront, search for the newly created category-level store keeping unit (SKU). It does not appear in the results.

Search is configured by default to return products, and not items, which is shown in the trace as the filter query: `fq=-catenttype_id_ntk_cs:ItemBean`. Category-level SKUs do not have any associated products, only items.

The best solution is to create an associated product bean for this item bean, rather than have item beans display. Products have a one-to-many relationship to items.

For example, assume that `shirt X` is available on your storefront. This shirt is available in three sizes and two colors. Therefore, this shirt has one *product* entry and six *item* entries. When searching for this shirt in the storefront, if displaying only product beans, the search appears one time in the search results. However, if displaying item beans, the shirt displays six times, one time for each item bean.

Alternatively, you can configure the search type to include SKUs or item beans in the search results, or you can configure the search type to show only SKUs (item beans) exclusively.

Navigate to the file: `Stores.war/<Your store name>/Snippets/Search/SearchSetup.jspf`.

Locate the section that is shown in Example 5-10.

Example 5-10 SearchSetup snippet section

```
<wcf:getData
type="com.ibm.commerce.catalog.facade.datatypes.CatalogNavigationViewType"
var="catalogNavigationView"
        expressionBuilder="${navigationView}" scope="request"
varShowVerb="showCatalogNavigationView"
        maxItems="${pageSize}" recordSetStartNumber="${WCParam.beginIndex}"
scope="request">
        <wcf:param name="searchProfile" value="${searchProfile}" />
        <wcf:param name="searchTerm" value="${newSearchTerm}" />
        <wcf:param name="searchType" value="${searchType}" />
```

The field `searchType` can have the following possible meanings:

- `0`: ANY (exclude SKU)
- `1`: EXACT (exclude SKU)
- `2`: ALL (exclude SKU)
- `3`: NONE (exclude SKU)
- `10`: ANY (include SKU)
- `11`: EXACT (include SKU)
- `12`: ALL (include SKU)
- `13`: NONE (include SKU)
- `100`: ANY (only SKU)
- `101`: EXACT (only SKU)
- `102`: ALL (only SKU)
- `103`: NONE (only SKU)

EXACT, NONE, ALL, and ANY can be defined, for example, by two search terms *t1* and *t2*, against two fields *f1* and *f2*:

- `EXACT +(f1:"t1 t2" f2:"t1 t2")`
- `NONE -(f1:(t1 t2) f2:(t1 t2))`

- `ALL +(f1:(+t1 +t2) f2:(+t1 +t2))`
- `ANY +(f1:(t1 t2) f2:(t1 t2))`

For example, to make ANY of the search terms entered return exclusively SKUs, set the search type to `100`:

```
<wcf:param name="searchType" value="100" />
```

Setting the search type to `100` changes the filter query from `fq=-catenttype_id_ntk_cs:ItemBean` to `fq=+catenttype_id_ntk_cs:ItemBean`.

Sales categories do not display in e-site stores

Perform the following steps:

1. In the Management Center Catalog tool, select **MadisonsEsite**, **Create Sales Catalog**, and **Sales** categories.
2. Link several products from the master catalog to the sales categories.
3. Build the full index.
4. Navigate to the storefront, with the sales catalog specified.

The sales categories are not displayed. This issue is resolved with authorized program analysis report (APAR) JR38606. You can download this APAR from Fix Central, or you can request this APAR from IBM Support. Note the additional steps that are listed in the APARs `readme.html` file when installing this APAR. After the APAR is installed, make sure to re-index the catalog.

Minor Catalog Entry changes no longer updated automatically in the search index

When you make minor updates, such as a price change, to the catalog, the predefined behavior is to process these updates automatically in the search index, to have the update show up in the storefront. (We discuss this process in greater detail in Chapter 4, “Search index life-cycle management” on page 93). The automatic update consists of two parts:

1. Processing the event as either delta or full, and adding this update to the TI_DELTA_CATENTRY table
2. Automatically synchronizing the index when the storefront is queried

For example, if a price change for catentry `12001` occurs, an event monitor picks up this change and determines that a delta update has occurred for catentry `12001`. An update of type `U` is made to the TI_DELTA_CATENTRY table. The next time that the store is queried, the run time notices that there are delta updates to process. These updates are made to the search index in a separate process.

By default, only delta updates are automatically indexed. If a full update has been identified by the event monitor and added to the TI_DELTA_CATENTRY table, this full update is not automatically re-indexed. Furthermore, if an update of type `F` is made to the TI_DELTA_CATENTRY table, further `U` updates are no longer added to this table. So, using the configuration that ships, having an `F` in the TI_DELTA_CATENTRY table stops further delta catentry changes from automatically updating the index.

To determine if this situation is why updates are not appearing automatically on your storefront, simply query the TI_DELTA_CATENTRY table. If you see an entry of type `F`, you can see what is blocking further delta catentry changes from automatically updating the index.

You have several possible resolutions:

- Simply run `di-preprocess` and `di-buildindex` manually. This action clears the TI_DELTA_CATENTRY table, and delta updates appear and are processed again.
- Modify the `wc-search.xml` file to get full index builds to also synchronize the index automatically. Navigate to the index section, and locate the line “`deltaUpdate="`**`true`**`" fullBuild="`**`false`**`">”`. Set `fullBuild=”true”`. Depending on how often large changes are made to the catalog, this solution can cause full re-indexing to run more often than desired.

Another possible reason changes might not appear on the storefront is caching. Query the Solr server directly to see if the expected updates appear in the query set. If the expected results are returned, the issue is outside the search index (and is likely a problem with caching). See the following Websphere Commerce Information Center topic to learn how to set up cache invalidation:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.admin.doc/concepts/cdc_cacheinv.htm

6

Search environment setup

This chapter describes in detail how to enable the search features of WebSphere Commerce V7.0 Feature Pack 2. This chapter also explains how to set up the WebSphere Commerce search environment by using an advanced configuration that is suitable for a production environment.

First, we describe in detail how to enable WebSphere Commerce V7.0 Feature Pack 2 search features on a local Commerce machine.

Then, we explain how to use an advanced deployment model to deploy the search on a remote search server.

At the end of the chapter, we explain several advanced scenarios of the search deployment related to clients' needs, the number of master catalogs, and multiple languages.

6.1 WebSphere Commerce search deployment overview

IBM WebSphere Commerce V7.0 Feature Pack 2 delivers customer-centric commerce with integrated search.

WebSphere Commerce search runs as a search server for your starter store that helps deliver targeted search results, while enabling customers to find products more effectively. When a shopper enters the search term, it is passed into the WebSphere Commerce integrated search framework. The search framework communicates with the WebSphere Commerce precision marketing engine to identify any business rules to apply to the query. The search query is then run using the search engine. There are many advantages to using the search engine over a traditional database search. We describe the advantages in this chapter and the following chapters.

The architecture of the search framework allows additional search adapters or vendor search engines to be used. We describe using the open source *Apache Solr*[1] search engine.

WebSphere Commerce search provides advanced administration features, such as dynamic clustering, database integration, rich document handling, distributed search, and index replication.

This chapter describes the distributed search model that is based on the Apache Solr search engine. Solr is a Java application that runs as a stand-alone full-text search server within a servlet container. Solr is built on the Lucene search library that is used for indexing and search functions and the Representational State Transfer (REST)-like API that is responsible for communication. Solr is easy to install and configure in any environment and configuration. Solr comes with an HTTP-based administrative interface.

In our examples, we use WebSphere Application Server V7.0 to deploy the `solr.war` file on a separate search machine, and we configure WebSphere Commerce to communicate with this search server.

WebSphere Commerce search works with the *search index* to provide search results that consist of structured and unstructured content, and external data. Search index is a large flat table that contains searchable fields that are optimized for search performance. The search index must be built before it can be used for any searches.

Deploying WebSphere Commerce search involves deploying the search server run time and setting up the search index structure for a specific master catalog. Typically, deploying the search run time is performed only one time, during feature enablement. Setting up the index for specific master catalogs is run when introducing a new master catalog that you want indexed.

6.1.1 WebSphere Commerce environment and search configuration

Figure 6-1 on page 117 and Figure 6-2 on page 117 show a simplified view of the hardware and software products and components that are used in our search examples.

[1] http://lucene.apache.org/solr/#intro

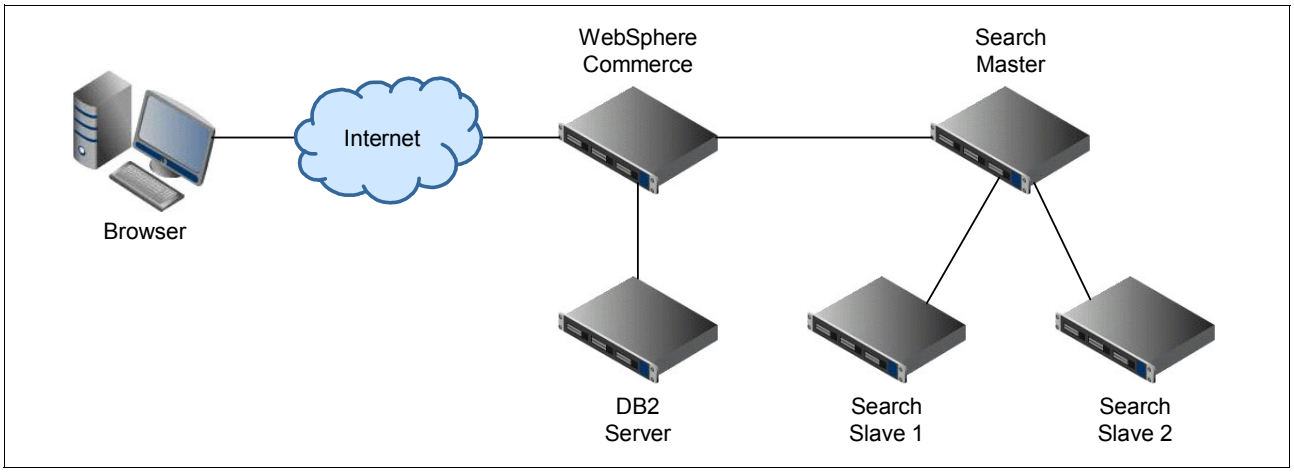


Figure 6-1 Hardware configuration with separate physical servers

To present the full power and advanced capabilities of the WebSphere Commerce V7.0 Feature Pack 2 search engine, we used an advanced hardware configuration and search servers on separate physical machines.

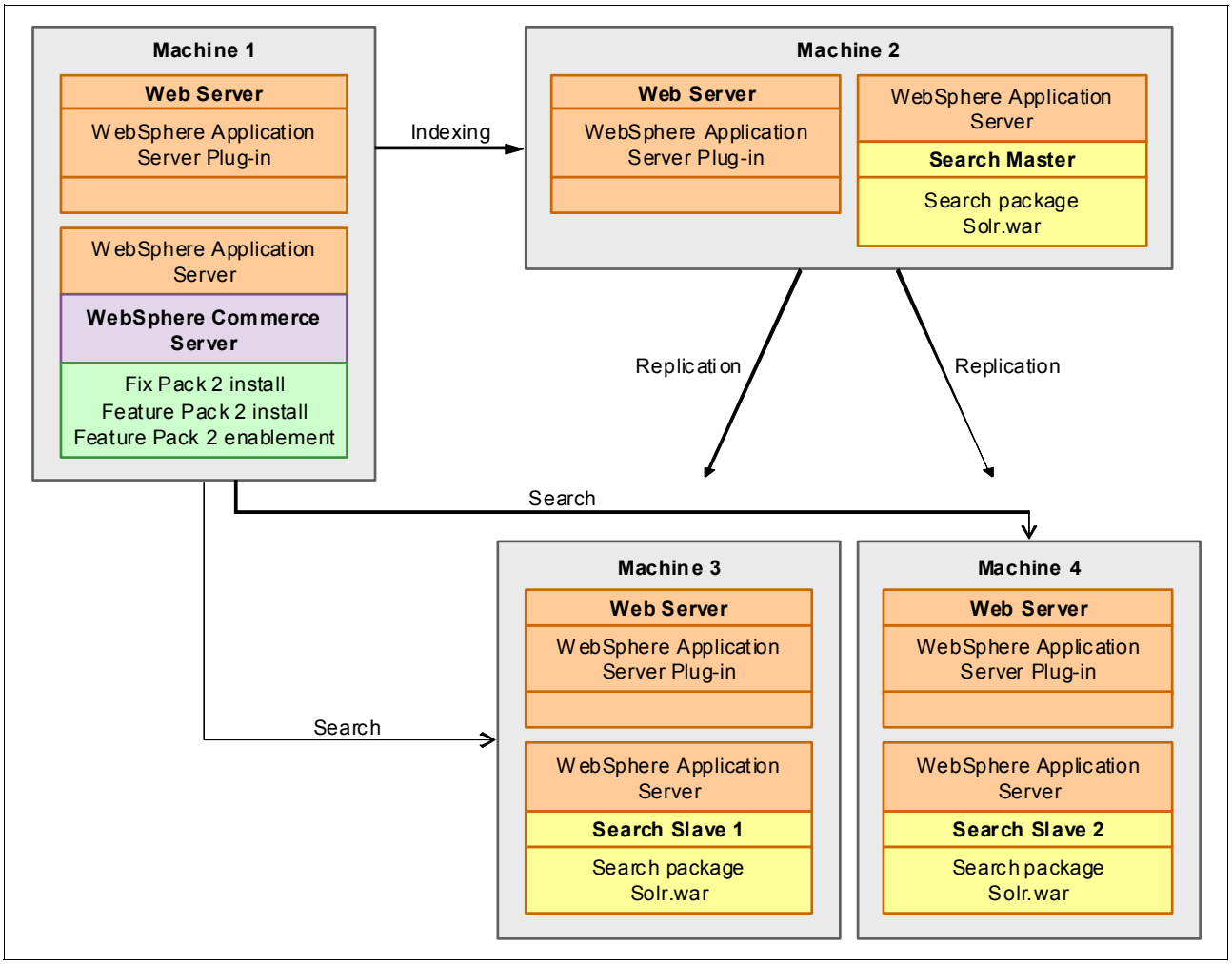


Figure 6-2 Software products and components

Figure 6-2 on page 117 shows the software components that are related to the WebSphere Commerce product, WebSphere Application Server product, and Apache Solr product.

The web server is the first point of contact with the customer and the incoming HTTP search requests for the e-commerce application. To interface efficiently with the WebSphere Application Server, the web server uses the WebSphere Application Server plug-in to manage the connections between the two components.

The WebSphere Commerce Server runs within the WebSphere Application Server. The database server holds the data of most of the applications, including the catalog data, product data, and customer data.

In our examples, we use the advanced (remote) search configuration, which means that the search servers are not on the same machine as the Commerce server. Each search server resides on a separate machine and runs in its own profile within the WebSphere Application Server.

6.1.2 Deployment options

The Apache Solr search platform does not come with its own deployment. To simplify the deployment of the application's `solr.war` file, WebSphere Commerce V7.0 Feature Pack 2 provides scripts that assist with the deployment.

Before deploying your search index, you must consider the available deployment options. There are three available deployment options for the WebSphere Commerce search solution:

- *Embedded mode* (WebSphere Commerce Developer only). WebSphere Commerce Server and WebSphere Commerce search run on a *single* machine and in the same Java virtual machine (JVM). The WebSphere Commerce search WAR (`solr.war`) becomes part of the WebSphere Commerce EAR file (`wc.ear`) that is deployed on the test server.
- *Standard (local) configuration*. WebSphere Commerce server and WebSphere Commerce search are deployed on a single machine, but they use *separate JVMs*. In order to function normally, additional memory is required. This configuration is suitable for small sites with low traffic volumes. In this case, Solr is configured to run in another profile of an existing application server. Similar to the embedded mode, you can configure the standard local configuration option automatically during the feature enablement.
- *Advanced configuration*. WebSphere Commerce Server and WebSphere Commerce search run on *separate machines*. Solr is deployed on its own WebSphere Application Server, in a separate profile, which provides greater flexibility and scalability. Scripts are provided to assist in the manual deployment. *Most companies use this configuration for a production environment*. The WebSphere Application Server cluster, with multiple WebSphere Commerce server and WebSphere Commerce search nodes, is deployed for scalability and failover support. We describe this configuration in detail in this chapter.

6.1.3 Deploying the search server

The first step is to deploy the search application, which is the `solr.war` file, which, for the embedded and standard deployments, is included in the WebSphere Commerce *foundation feature* enablement.

The WebSphere Commerce foundation feature adds new groups of services to the WebSphere Commerce Server. The WebSphere Commerce foundation feature is required when enabling features after installing a WebSphere Commerce Feature Pack. Enabling the WebSphere Commerce foundation ensures that the enabled features can function correctly

with WebSphere Commerce. It provides the groups of services that are required to support the new functionality.

The advanced configuration deployment is a manual process with provided scripts, which we describe in detail in this chapter. The next step for the standard and advanced deployments is to configure the web server for the Solr application. WebSphere Commerce must be able to send search requests to the search server. We describe the installation and usage of the IBM HTTP server, and we describe the installation and updating of the WebSphere Application Server V7.0 for this purpose later in this chapter.

6.1.4 Deployment illustrations

Figure 6-3 illustrates deploying the WebSphere Commerce search in the advanced configuration, where the WebSphere Commerce Server and WebSphere Commerce search e run on separate machines.

For more information, see the information center pages for administering WebSphere Commerce search and deploying WebSphere Commerce search:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdmanagesearch.htm

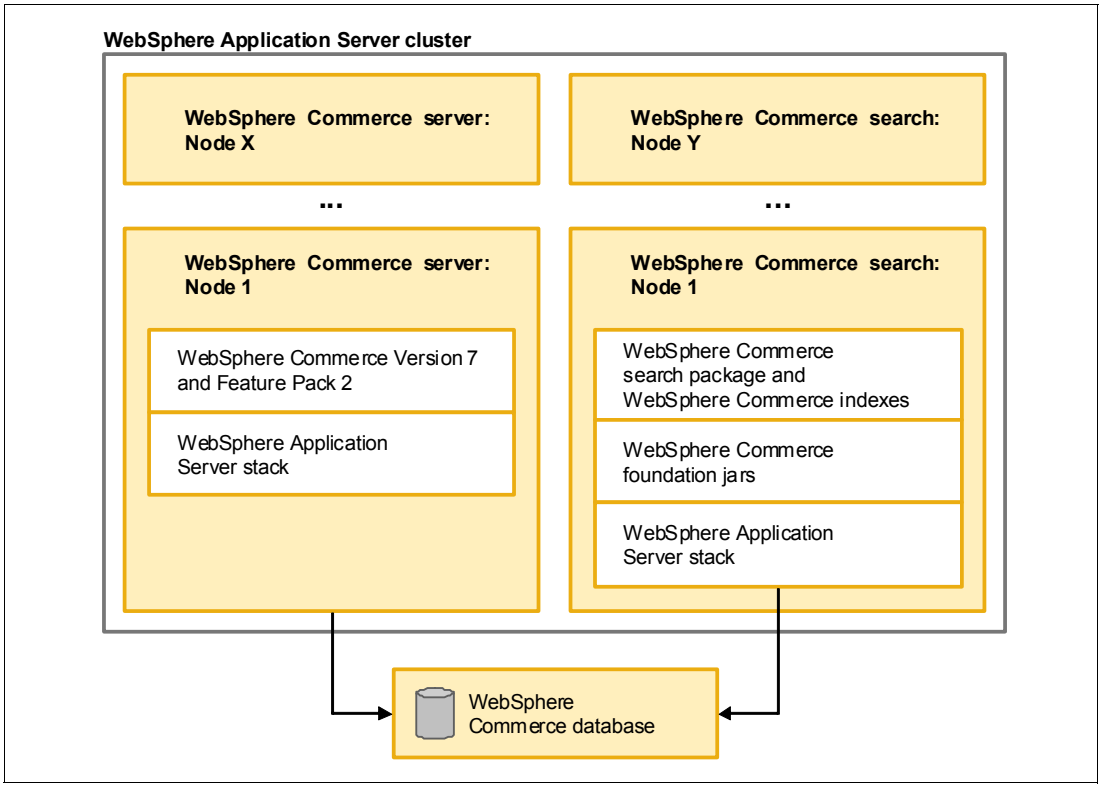


Figure 6-3 Deploying WebSphere Commerce search in the advanced configuration

The flow chart in Figure 6-4 on page 120 describe the flow to set up the standard and advanced search configurations.

The flow includes two major steps:

1. Deploying the WebSphere Commerce search server and deploying the search on WebSphere Application Server.

2. Setting up the WebSphere Commerce search index structure. After the search server is deployed in WebSphere Application Server, you can use the `setupSearchIndex` utility to set up the search index structure. You configure the WebSphere Commerce server and create the Solr home and Solr core structure for a specific master catalog.

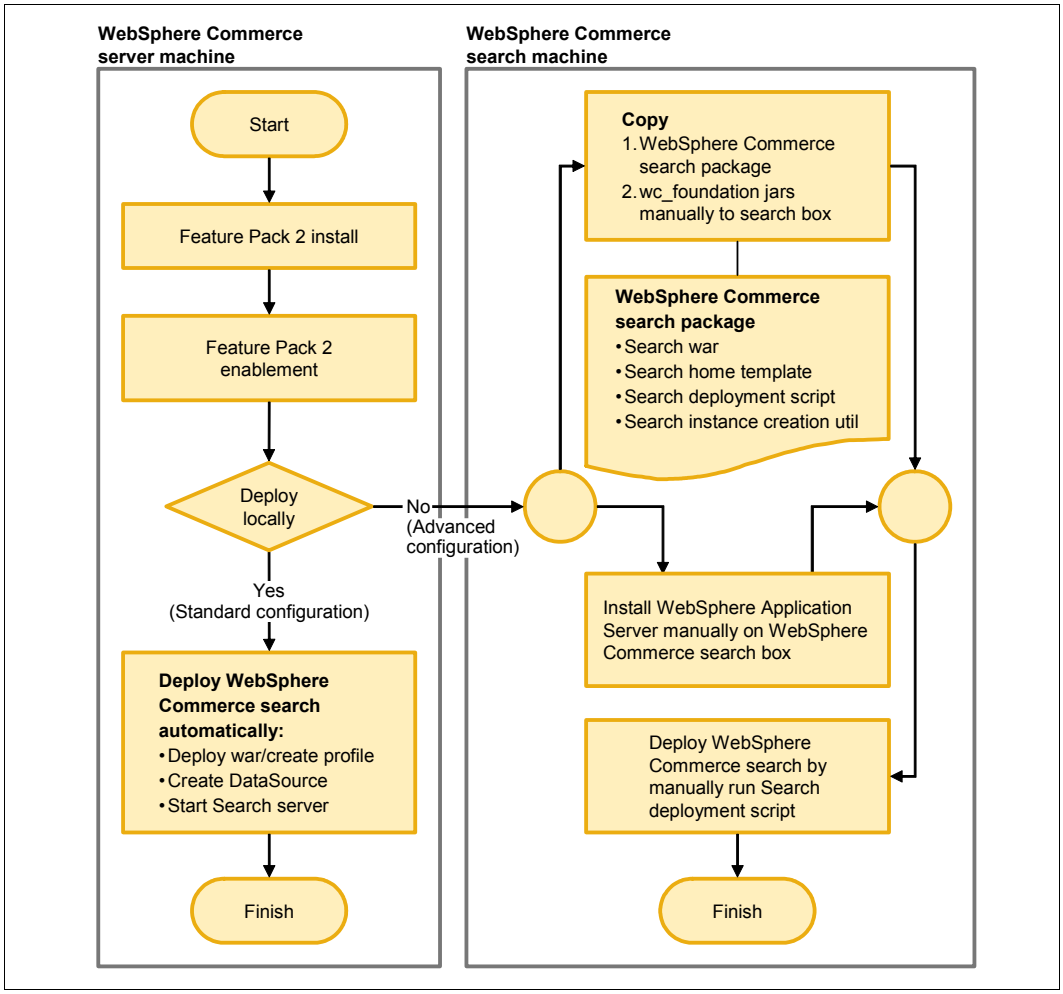


Figure 6-4 Standard and advanced configuration setup flow

6.1.5 Publishing the store after the search deployment

After deploying WebSphere Commerce search, the next step is to publish the store. The `enhancements.sar` file that is provided with WebSphere Commerce V7.0 Feature Pack 2 contains examples of the storefront code that is needed to call the catalog search services. After the store is published, it is important to document the masterCatalogId that was used.

The `setupSearchIndex.sh` script that is provided with WebSphere Commerce V7.0 Feature Pack 2 identifies the languages that are supported by the store and creates each required index. A search index is created for each combination of master catalog and supported language. With the advanced deployment option that is described in this chapter, creating the search index is a two-part process. The `setupSearchIndex.sh` script is run on both the WebSphere Commerce server and the search server using separate parameters.

You must enable the Search-Based Navigation option for the store. The store function selection option is enabled by using the Store Management tool, which is located in the Catalog tab.

6.2 Deploying the WebSphere Commerce search using the advanced configuration

In the following sections, we describe in detail the complete deployment process for the WebSphere Commerce integrated search. This process is suitable for a real production environment. This configuration can greatly improve search performance. It also provides flexibility and makes maintenance much easier.

6.2.1 Deployment procedure overview

To complete the deployment procedure, you must perform all the following activities:

- Preparing the local WebSphere Commerce machine and feature enablement
- Preparing the remote search machine
- Creating the mappings from the WebSphere Commerce server to the remote server for search term associations
- Testing the search deployment

6.3 Preparing the local WebSphere Commerce machine and feature enablement

The installation of WebSphere Commerce V7.0 is the first step in a multiple-step procedure that has to be performed in order to fully enable the WebSphere Commerce V7.0 Feature Pack 2 features, and to prepare the environment for advanced search. Refer to the installation plan in the following section to prepare your environment.

6.3.1 Installing and updating the plan for WebSphere Commerce V7.0 Server and its components

This section provides information to help you plan your configuration of WebSphere Commerce V7.0. We provide the software update procedures and necessary download links to complete the configuration and feature enablement of the WebSphere Commerce V7.0 Feature Pack 2 server for the advanced search model.

We used the SUSE Linux Enterprise Server 11.4 64-bit operating system on all machines in our environment. We installed the WebSphere Commerce V7.0 product from the appropriate compact disc or distribution media.

WebSphere Application Server V7.0 product updates

You can obtain the details about the preparation of the Linux operating system for the WebSphere Application Server V7.0 product installation at this website:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.nd.doc/info/ae/ae/tins_prepare.html

After installation, the first part of the update plan relates to WebSphere Application Server V7.0. After you update the WebSphere Application Server V7.0 Installer, you use this updated installer to update the WebSphere Application Server, plug-ins, HTTP IBM Server (HIS), and Java software development kit (SDK). You can obtain the download help links and instructions for the WebSphere Application Server V7.0 updates at this website:

http://www-01.ibm.com/support/docview.wss?uid=swg24029073

Installing WebSphere Application Server, creating a default profile, and updating the server to the latest version

Refer to 6.5, “Installing WebSphere Application Server with default profile creation and update” on page 137.

WebSphere Commerce V7.0 product updates and FEP2 enablement

The second part of the update plan relates to WebSphere Commerce V7.0 Feature Pack 2.

Perform the following steps to update your version of WebSphere Commerce V7.0 and enable Feature Pack 2:

1. Update WebSphere Commerce Installer to WebSphere Commerce Fix Pack 2 (7.0.0.2) for Server:
   - Download the link:

     ftp://public.dhe.ibm.com/software/websphere/commerce/70/updi7002/download.updii.7002.linux.amd64.zip
   - Refer to this Help link:

     http://www-01.ibm.com/support/docview.wss?uid=swg24013502

   You use the WebSphere Commerce Update Installer tool to apply fix packs and authorized program analysis reports (APARs) to your WebSphere Commerce installation. The WebSphere Commerce Update Installer is packaged as a separate program. You must first install the Update Installer before you can apply the WebSphere Commerce maintenance packages (`.pak` files) to your WebSphere Commerce configuration.
2. Install the WebSphere Commerce V7.0.0.2 Fix Pack:
   - Download this link:

     ftp://delivery04-mul.dhe.ibm.com/ecc/hsb/H36769802/7.0.0-WS-WCServer-FP002.pak

   A fix pack is applied on top of a specific version or release.
3. Create a WebSphere Commerce instance:
   - Create the WebSphere Commerce instance using Configuration Manager or ANT targets.
   - Refer to this Help link:

     http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.install.doc/tasks/tig_part_create_instances.htm

   In this step, we connect to the database. Ensure that you can connect to the DB2® server. After successfully creating an instance, we can install Feature Pack 2.
4. Install IBM WebSphere Commerce V7.0 Feature Pack 2.0 (FEP2) for Multiplatform Multilingual:
   - Download these instructions:

     Refer to IBM Passport Advantage® and locate the CZN2EML package.
5. Enable the Feature Pack 2 features:

   WebSphere Commerce V7.0 Feature Pack 2 includes all the features that were previously released with Feature Pack 1.0. The WebSphere Commerce V7.0 Feature Pack 2 Version 2.0.0.0 includes the following features:

- Foundation 7.0.2.0
- Management-center 7.0.2.0
- Store-enhancements 7.0.2.0
- Social-commerce 7.0.2.0
- Content-version 7.0.2.0
- Sterling-integration 7.0.2.0

After installing Feature Pack 2, you can choose which of these features you want to enable. You can enable several of these features automatically.

In the following two sections, we describe the enablement of the WebSphere Commerce *foundation feature* and *starter store-enhancements feature*.

Tip: You do not need to perform the foundation feature enablement for this procedure to succeed. Enabling the store enhancement features only automatically enables the dependent features, which are a part of the management-center and foundation features.

If you need to enable any other feature and if you understand the dependencies between the features and their automatic enablement, visit this information center page and prepare your environment, as suggested:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.install.doc/tasks/tigfepenable.htm

6.3.2 WebSphere Commerce foundation feature enablement procedure

Prepare your WebSphere Commerce machine for feature enablement. Perform the following procedure:

1. Log on as a WebSphere Commerce non-root user.
2. Navigate to the following directory:

   ```
   cd /opt/IBM/WebSPhere/AppServer/bin
   ```

3. Start the server1 instance:

   ```
   ./startServer server1
   ```

4. Increase the file handle limit.

   Switch to the root user and issue the command:

   ```
   ulimit -n 8192
   ```

   Switch back to the non-root user.

5. Run the enablement script for search WebSphere Commerce foundation feature.

   Enable the WebSphere Commerce foundation feature change working directory:

   ```
   cd /opt/IBM/WebSphere/CommerceServer70/bin
   ```

6. Execute the script in Example 6-1.

Example 6-1 Enable the WebSphere Commerce foundation feature

```
./config_ant.sh -buildfile
/opt/IBM/WebSphere/CommerceServer70/components/common/xml/enableFeature.xml
-DinstanceName=demo -DfeatureName=foundation -DdbUserPassword=inst1pas
-DremoteSearchEngine=true
```

We define the parameters in Example 6-1:

DinstanceName The name of the WebSphere Commerce instance.

DfeatureName The name of the FEP2 feature for enablement.

DdbUserPassword The password for the DB2 user connecting to the database.

DremoteSearchEngine To deploy the search server on a remote server, specify the value of this flag parameter to `true`. This value is necessary for our remote deployment configuration.

7. Ensure that the script runs successfully. If the script runs successfully, you see the `BUILD SUCCESSFUL` message displayed in the console ().

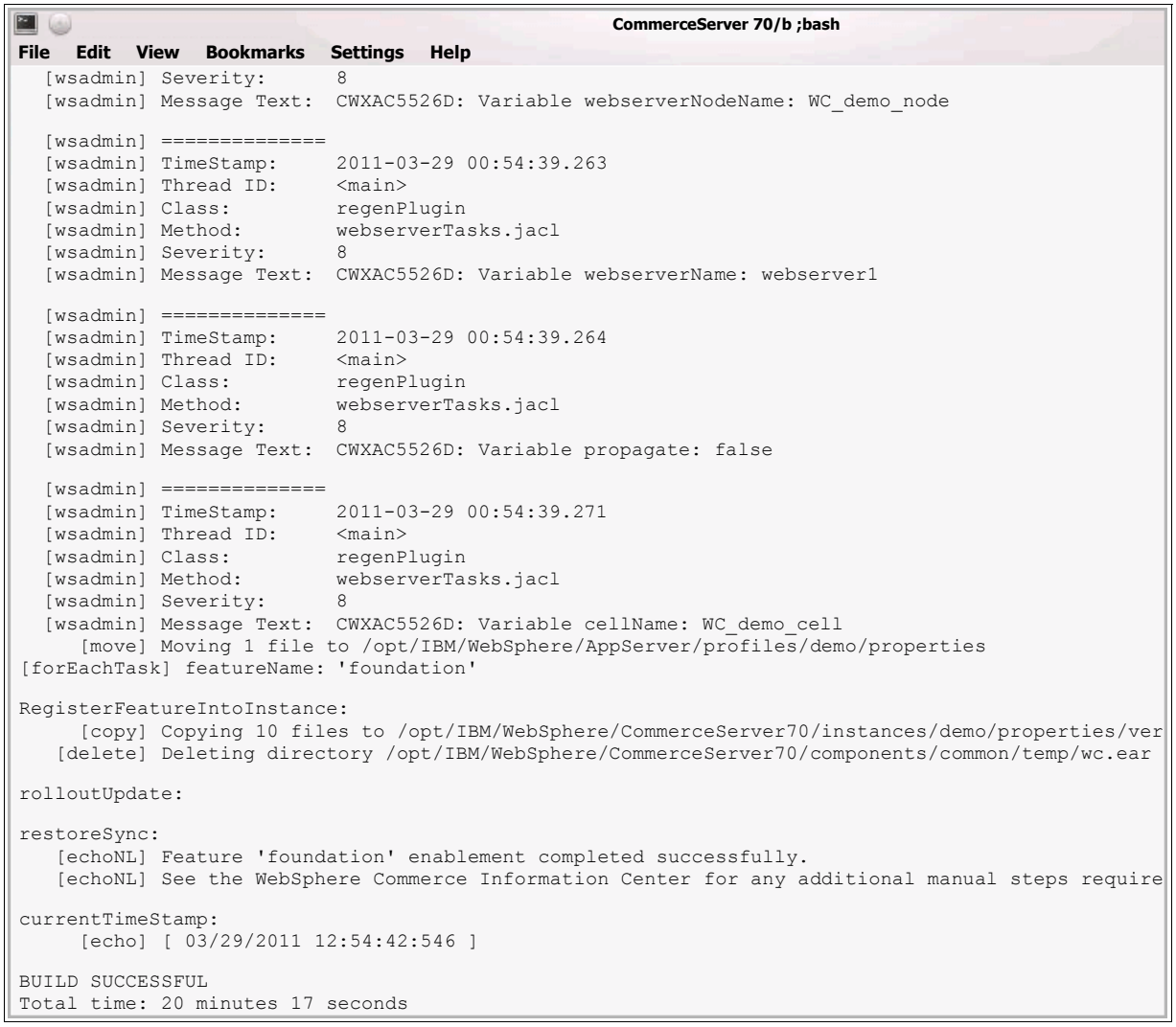


Figure 6-5 Console output for the execution of the config_ant.sh script

8. You can locate the complete log file at this location:

   `/opt/IBM/WebSphere/CommerceServer70/instance/demo/logs/enablefoundation_2011.03.29_00.34.27.557.log`

6.3.3 Madisons starter store enhancement enablement procedure

The Madisons starter store enhancements include co-shopping functionality, search integration, subscriptions and recurring orders, Coremetrics Intelligent Offer functionality, and support for Management Center price lists and price rules.

In this section, you complete the following steps:

1. Run the enablement script for starter store enhancements.
2. Publish the Madisons starter store archive.
3. Publish the Madisons enhancements store archive.
4. Enable the search-based navigation store function.

Running the enablement script for starter store enhancements

Perform these steps to enable WebSphere Commerce starter store enhancements in the Madisons store:

1. Ensure that you are logged on as the WebSphere Commerce non-root user.
2. Increase the file handle limit.

   Switch to the root user and issue the command:

   ```
   ulimit -n 8192
   ```

   Switch to the non-root user.
3. Run the enablement script.

   Running the starter store enhancements enablement script also enables the *Management Center* feature. If the Management Center feature is currently disabled, it is enabled after running the starter store enhancements enablement script.

   Use this command to enable the store-enhancements feature change working directory:

   ```
   cd /opt/IBM/WebSphere/CommerceServer70/bin
   ```

   Execute the script in Example 6-2.

   Example 6-2 Enabling store-enhancements feature

   ```
   ./config_ant.sh -buildfile
   /opt/IBM/WebSphere/CommerceServer70/components/common/xml/enableFeature.xml
   -DinstanceName=demo -DfeatureName=store-enhancements -DdbUserPassword=inst1pas
   ```

4. Ensure that the script runs successfully. If the script runs successfully, the `BUILD SUCCESSFUL` message is displayed in the console (Figure 6-6 on page 126).

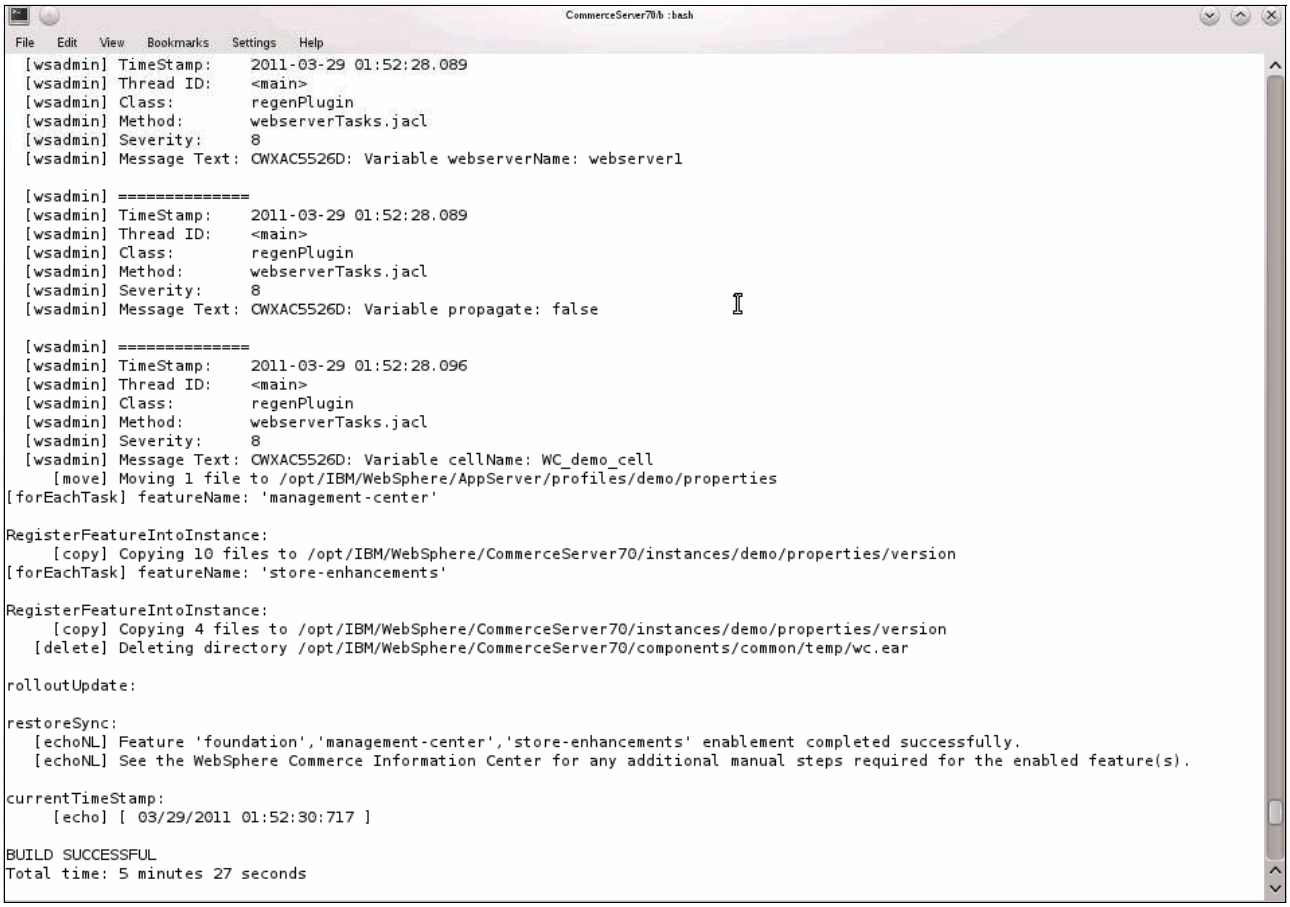


Figure 6-6 Console output for the execution of the script config_ant.sh

5. You can see the complete log file at this location:

   `/opt/IBM/WebSphere/CommerceServer70/instance/demo/logs/enablestore_enhancements_2011.03.29_01.47.05.179.log`

> **Tip:** There is no need to add the following parameters to the enablement script for our example, because the WebSphere Commerce foundation feature is enabled in previous step:
>
> `[-DremoteSearchEngine=true] [-DsolrHome=solrhome]`

Publishing the Madisons starter store archive

Use the publish wizard to publish the `Madisons-FEP.sar` store archive that was updated in Feature Pack 2.

Complete the following steps:

1. Open the Administration Console (Figure 6-7 on page 127). Enter the user name and password and click **Log On**.

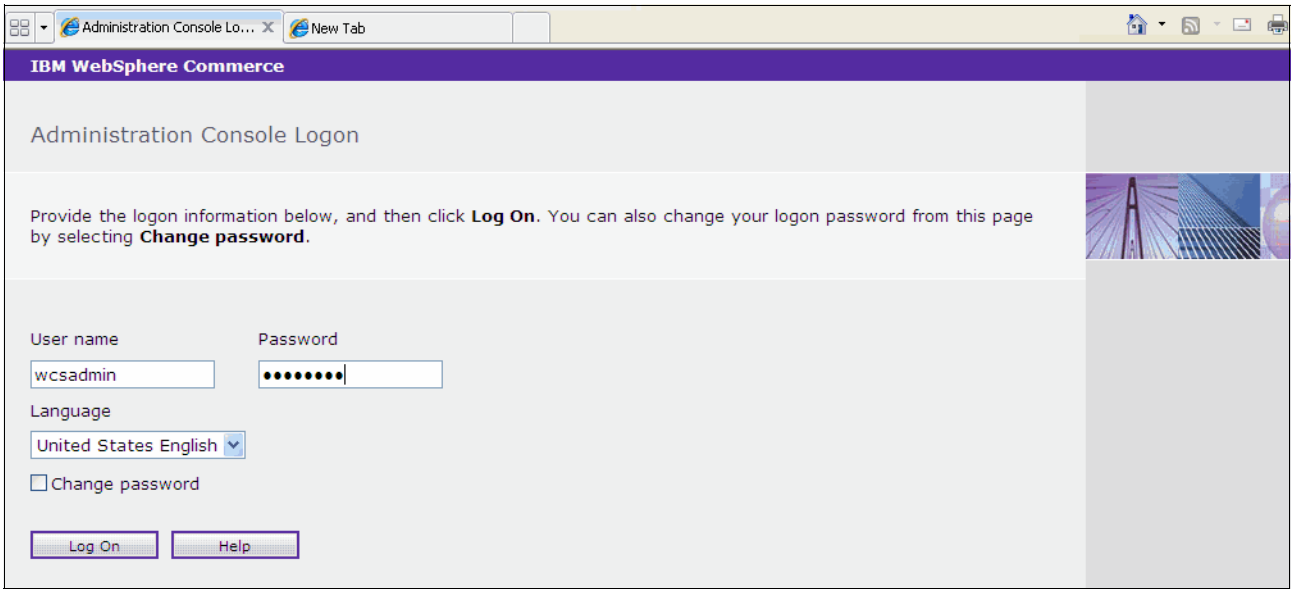


Figure 6-7 Administration Console Logon page

2. Select the **Site** option (Figure 6-8) and click **OK**.

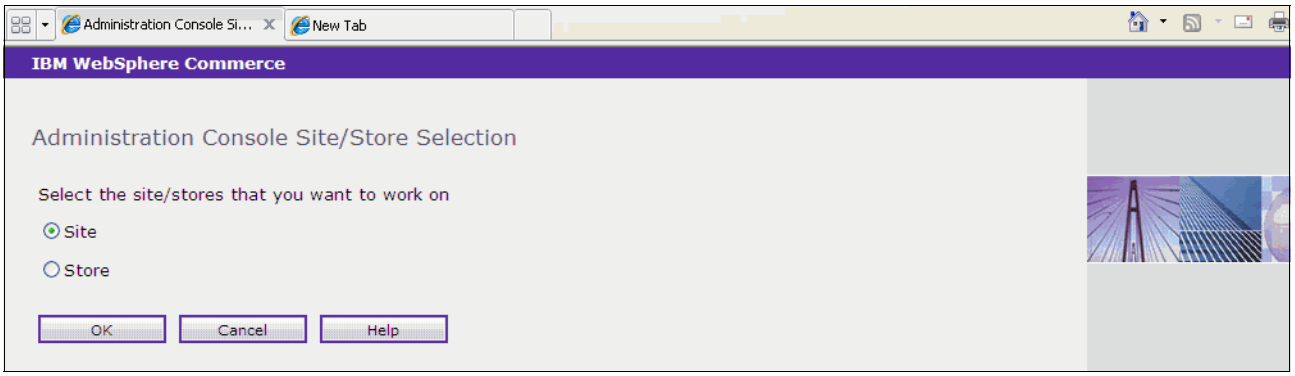


Figure 6-8 Administration Console: Selecting the site

3. Go to the Store Archives menu and select **Publish** (Figure 6-9).

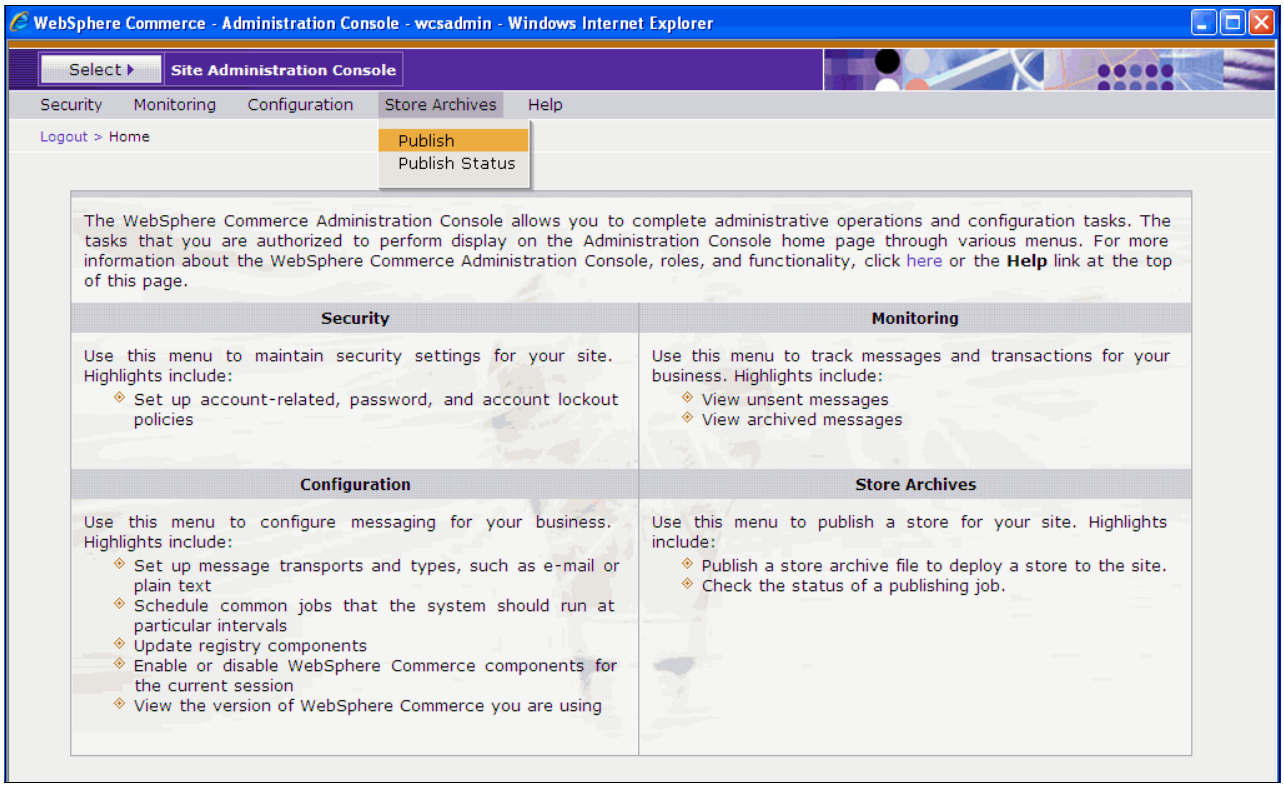


Figure 6-9 Selecting the Publish option from the Store Archives menu

The Store Archive page shows list of store archives that can be published.

4. Select **Madisons-FEP.sar** from the list (Figure 6-10).

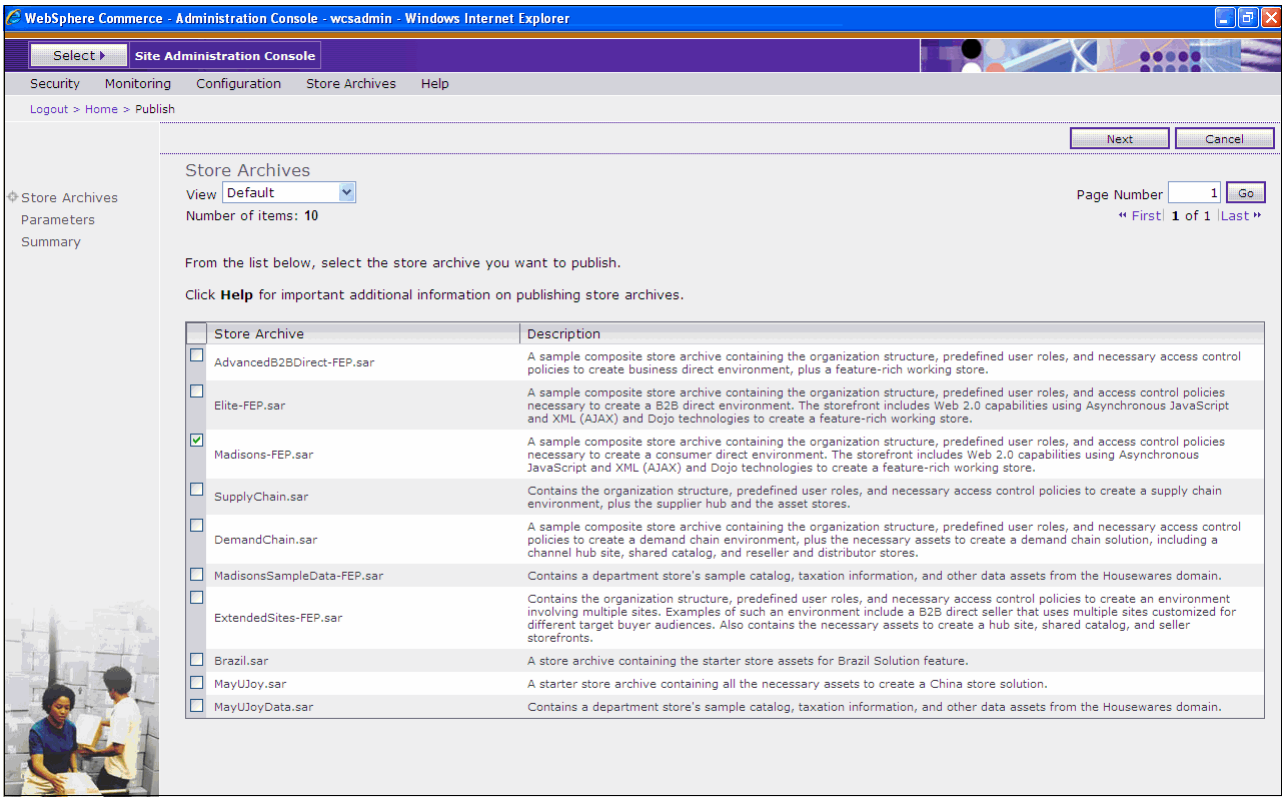


Figure 6-10 Selecting the Madisons-FEP.sar archive

5. Click **Next**. You see a window that is similar to Figure 6-11.

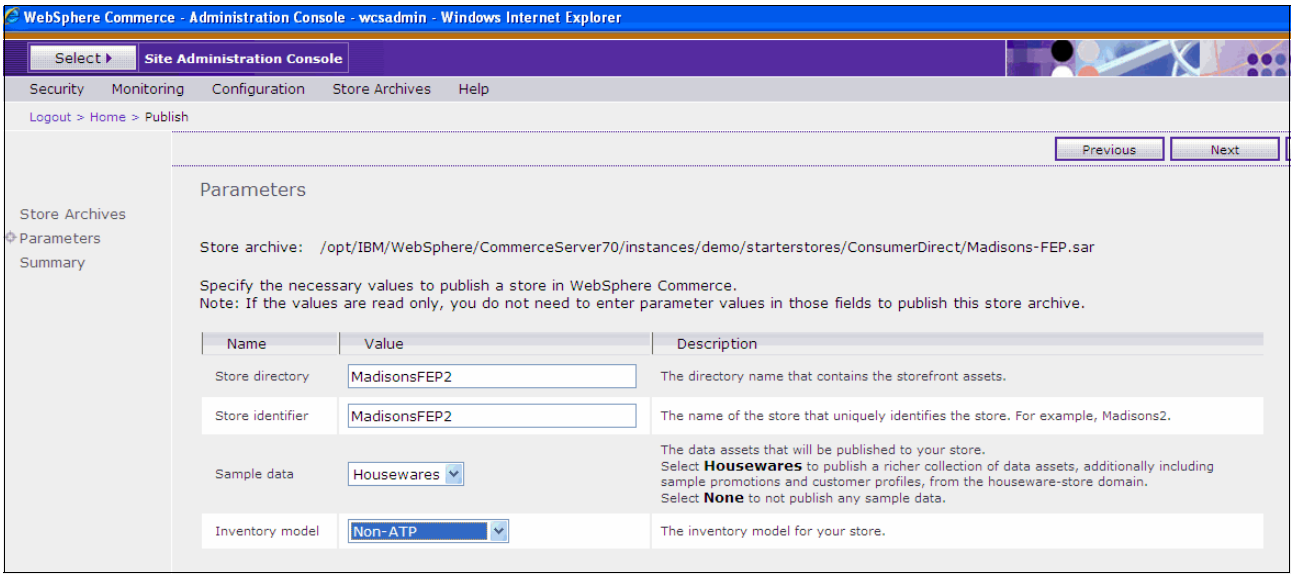


Figure 6-11 Parameters page

A page with parameters is displayed (Figure 6-11). Depending on the store archive that you selected, various parameters are displayed.

6. Click **Next**. You see a window similar to Figure 6-12.

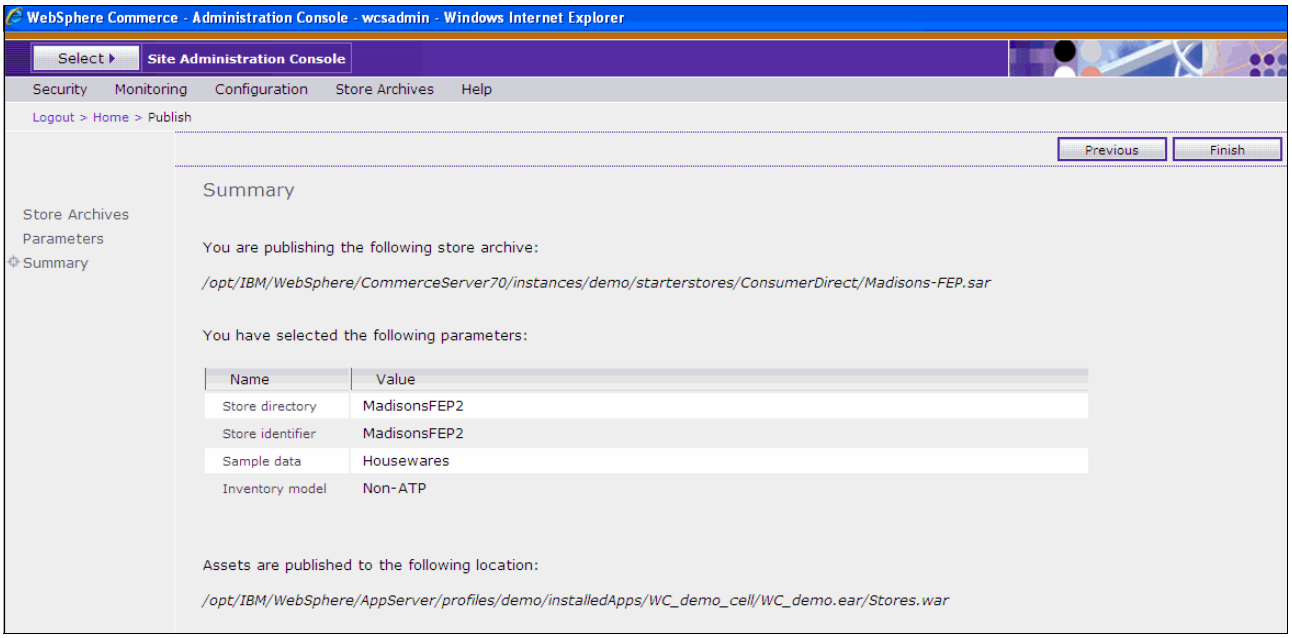


Figure 6-12 Summary page

The Summary page displays.

7. Click **Finish** and you see a window that is similar to Figure 6-13.

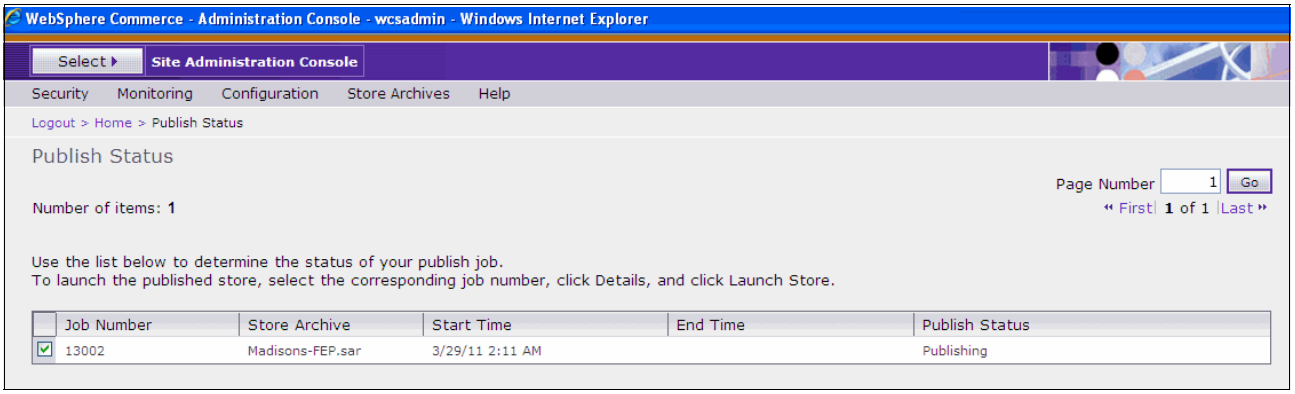


Figure 6-13 Publish status page

The Publish Status confirmation page displays with a job number.

8. Click **Refresh**.

   Verify that the field Publish Status displays `Successful` for its value (Figure 6-14).

You have successfully published the Madisons starter store archive.

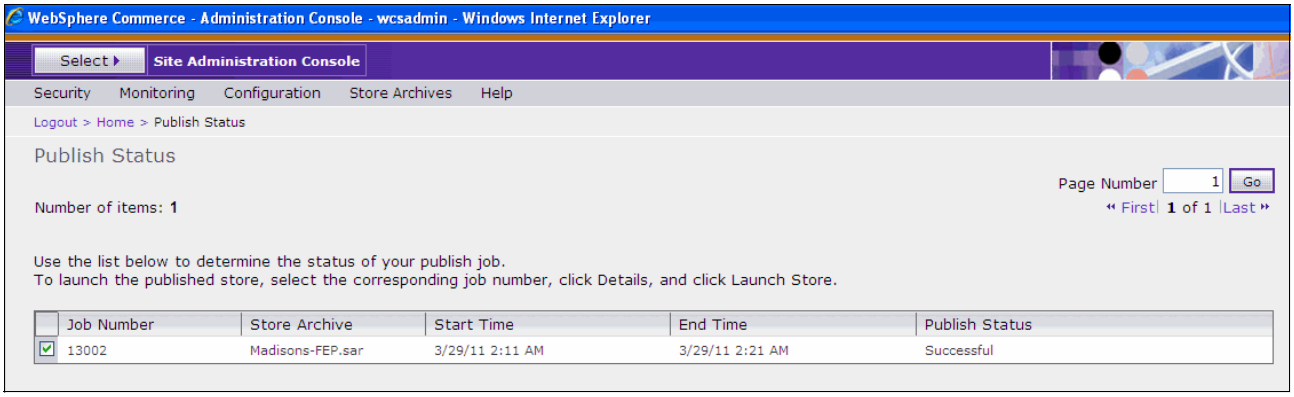


Figure 6-14 Publish Status page: Successful publishing

Publishing the Madisons enhancements store archive

The `MadisonsEnhancements.sar` file is a store archive that adds features to the Madisons starter store.

Here, we describe the steps of the Publish wizard in the Administration Console that are needed to publish this store archive on top of the Madisons starter store.

Complete the following steps:

1. Repeat steps 1 on page 126 and 2 on page 127 from the Publishing the Madisons starter store archive section for opening the Administrative Console and the Selecting the Site option.

2. From the menu Store Archives, select **Publish** (Figure 6-15).

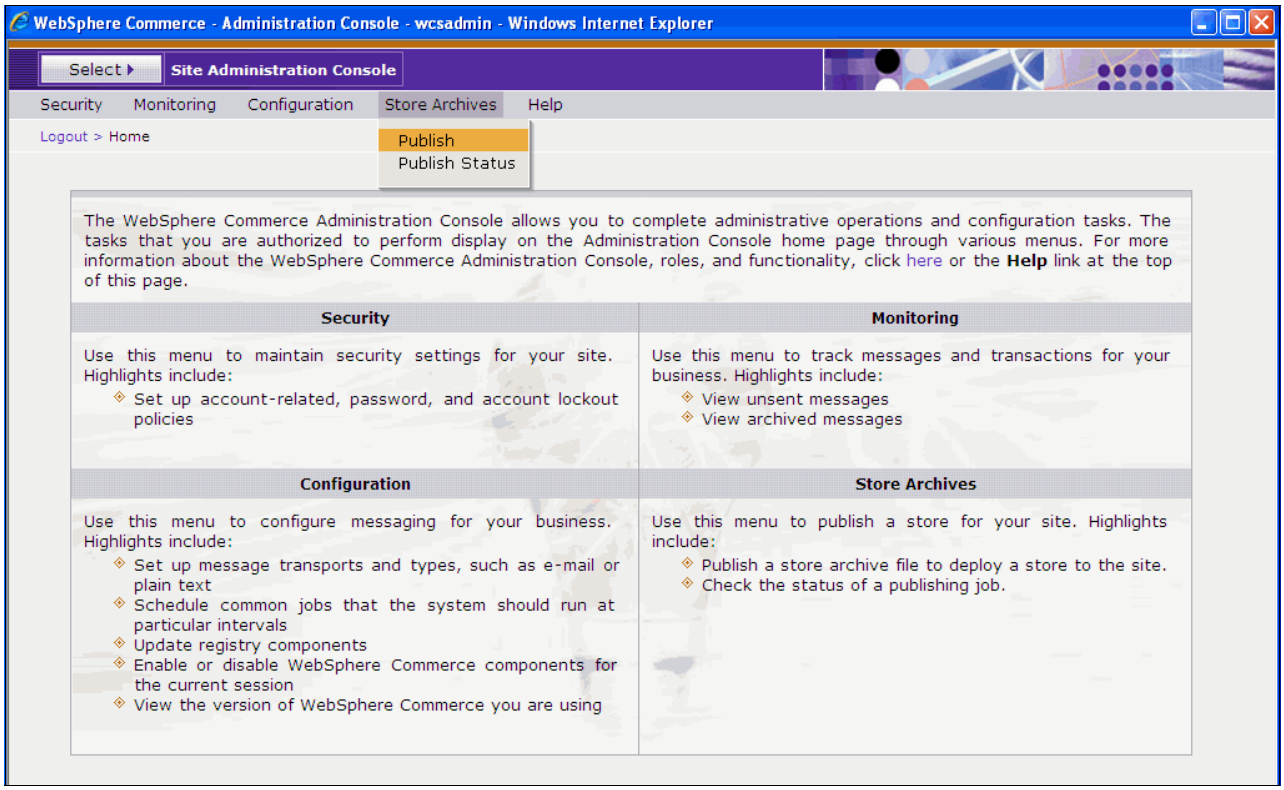


Figure 6-15 Selecting the Publish option from Store Archives menu

3. On that Store Archives page that is displayed, from the **View** list box, select **Add on Feature** (Figure 6-16).
4. Select **MadisonsEnhancements.sar** by clicking the check box field next to its name.

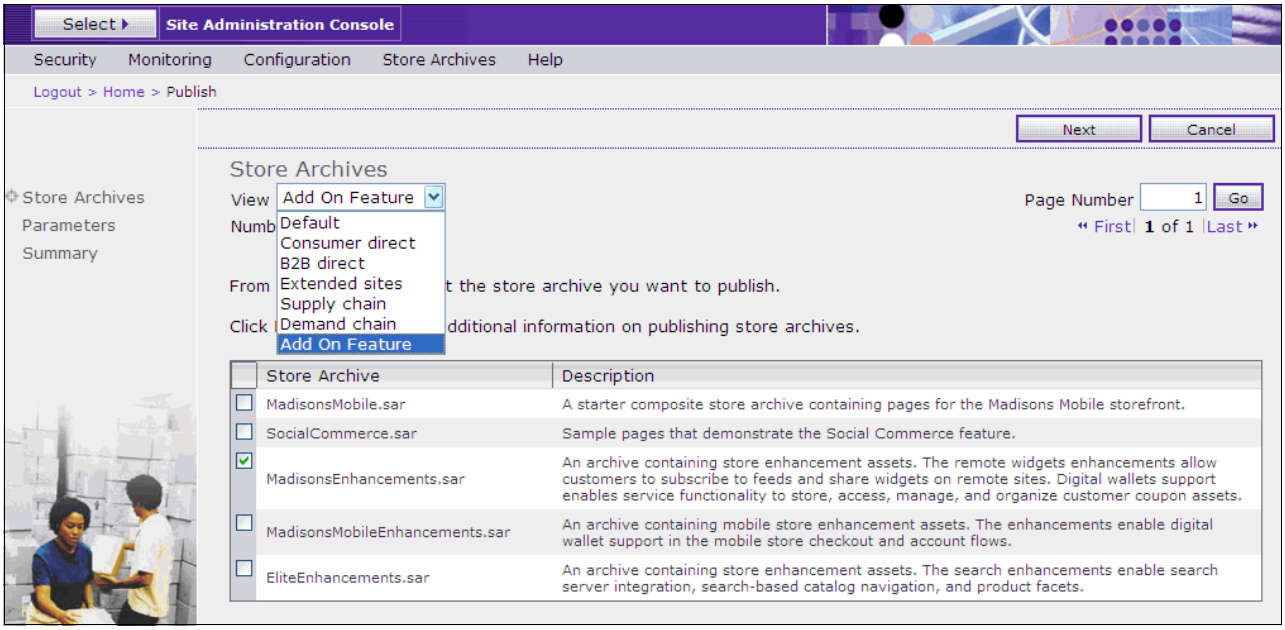


Figure 6-16 Selecting the Add On Feature from the View list box on the Store Archives page

5. Click **Next**.

The Publish Wizard publishes the Madisons enhancements store archive on top of the Madisons starter store. Continue performing the following steps:

1. On the Parameters page that is displayed, select the Madisons starter store (Marketing store identifier) called **MadisonsFEP2** (Figure 6-17).

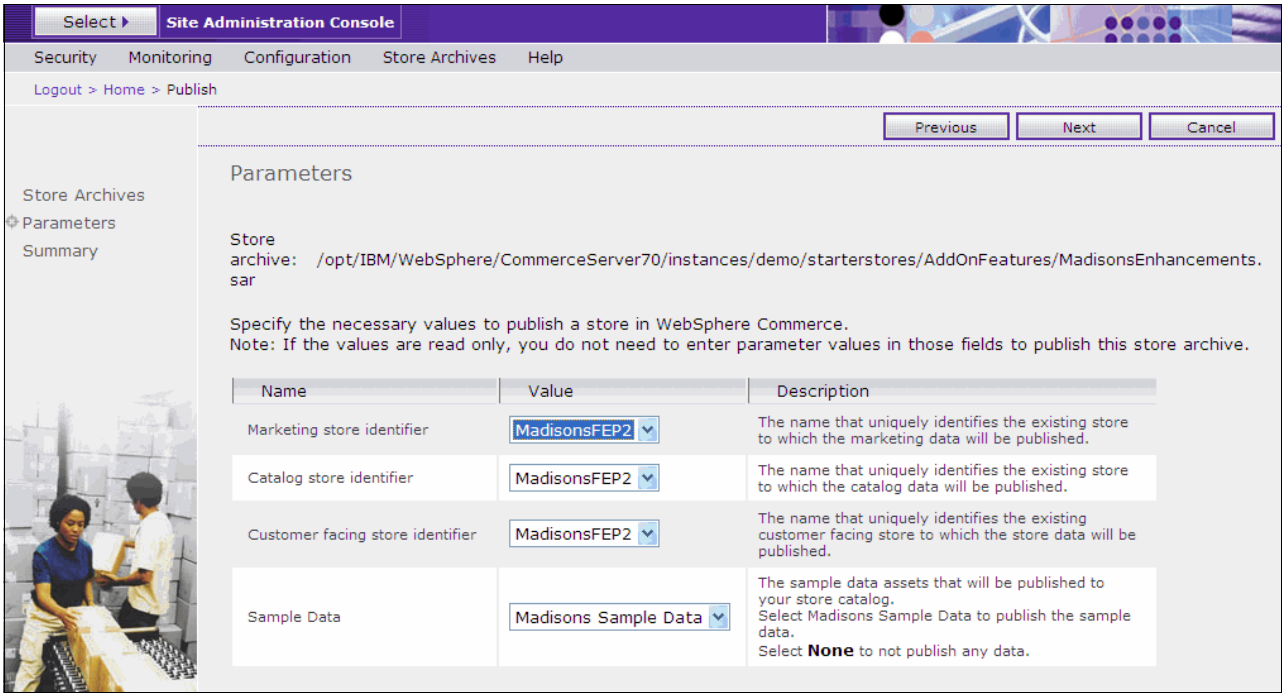


Figure 6-17 Parameters page

2. Click **Next**. You see a window similar to Figure 6-18.

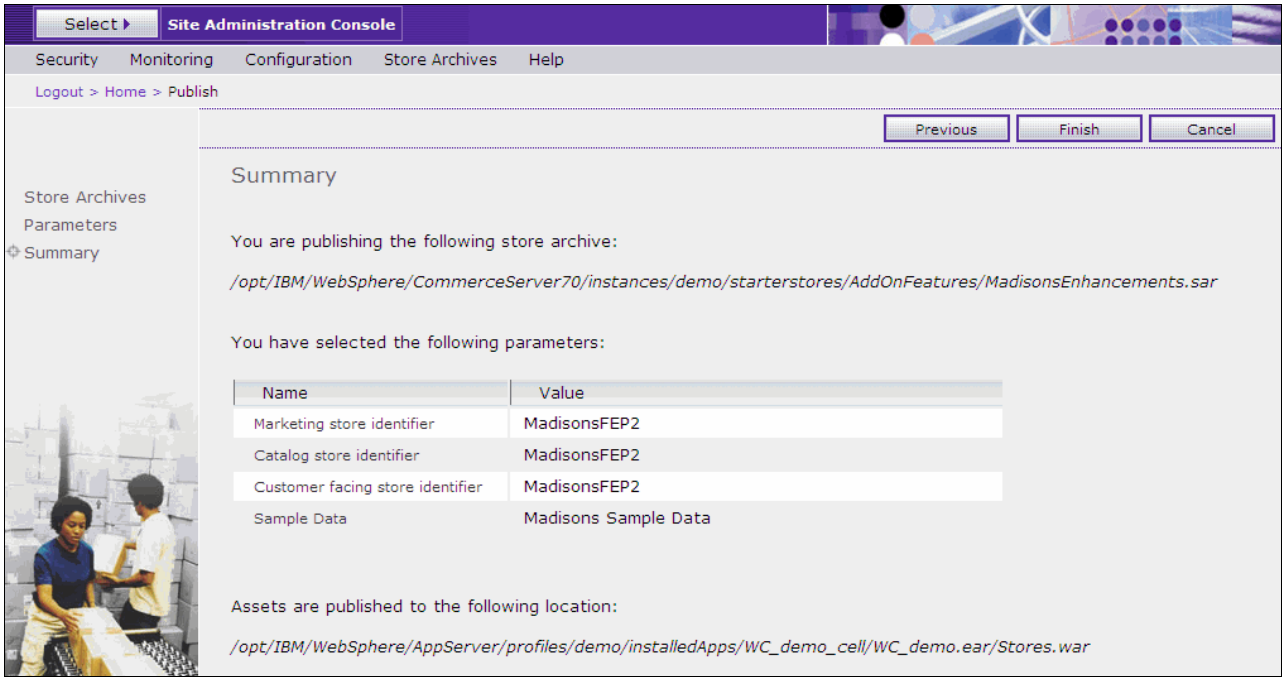


Figure 6-18 Summary page

3. The summary page displays a listing of the selected store archive parameters and the location to which the archive is to be published. Complete these tasks:
   a. Review the summary information and click **Finish**.
   b. In the Publish Status window, click **Refresh** to check the publishing status (Figure 6-19).

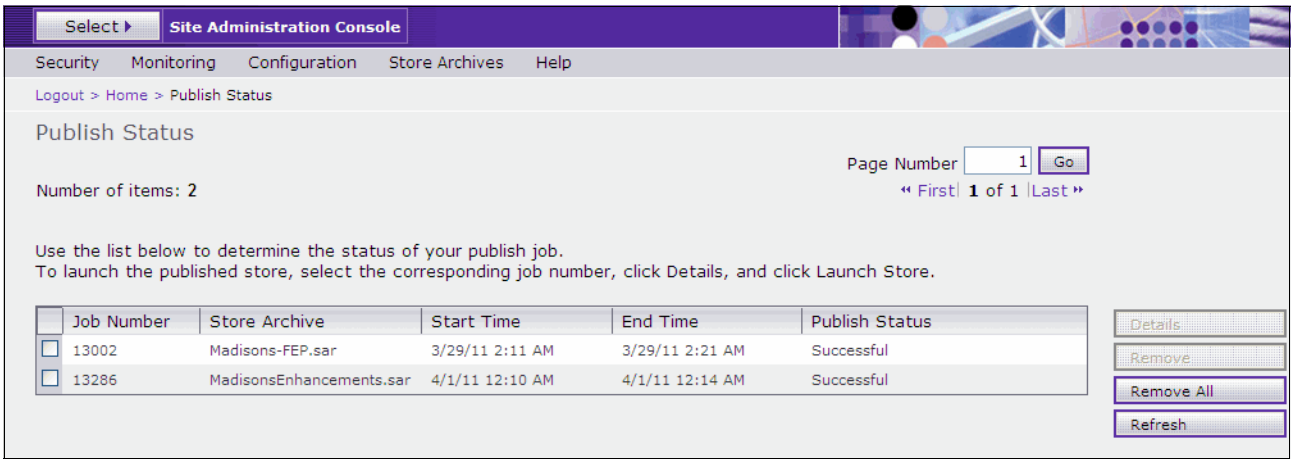


Figure 6-19 Publish Status page with successful publishing

4. Verify that the value of the field Publish Status is `Successful`.

You have successfully published the Madisons enhancements store archive.

Enabling the search-based navigation store function

This section describes how to enable the search-based navigation for a storefront.

Search-based navigation enhances starter stores by performing searches against categories when a shopper browses a storefront catalog. All immediate categories and products are returned in the results, and you can also include the related subcategories to increase the scope.

Complete the following steps to enable the catalog-based navigation:

1. Log on to the Management Center.

2. From the Management Center Tools drop-down menu, select **Store Management** (Figure 6-20).

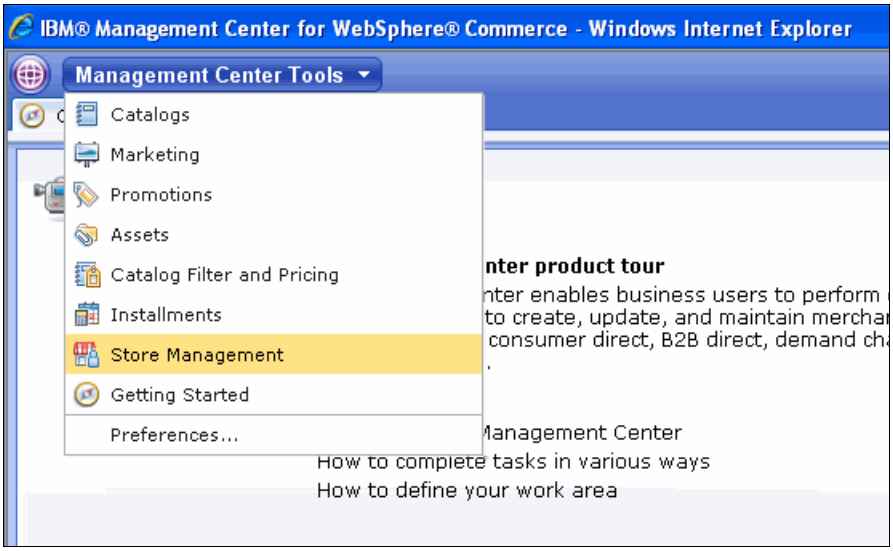


Figure 6-20 Store Management selection

3. Click **Stores** in the explorer view (Figure 6-21).

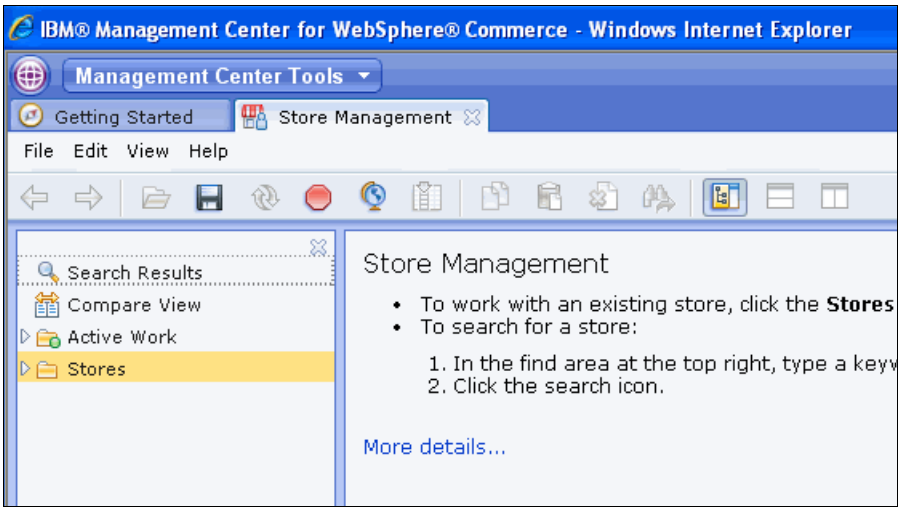


Figure 6-21 Selecting Stores in the explorer view

4. Right-click the displayed **MadisonsFEP2** store in the Stores - List and click **Open** (Figure 6-22).

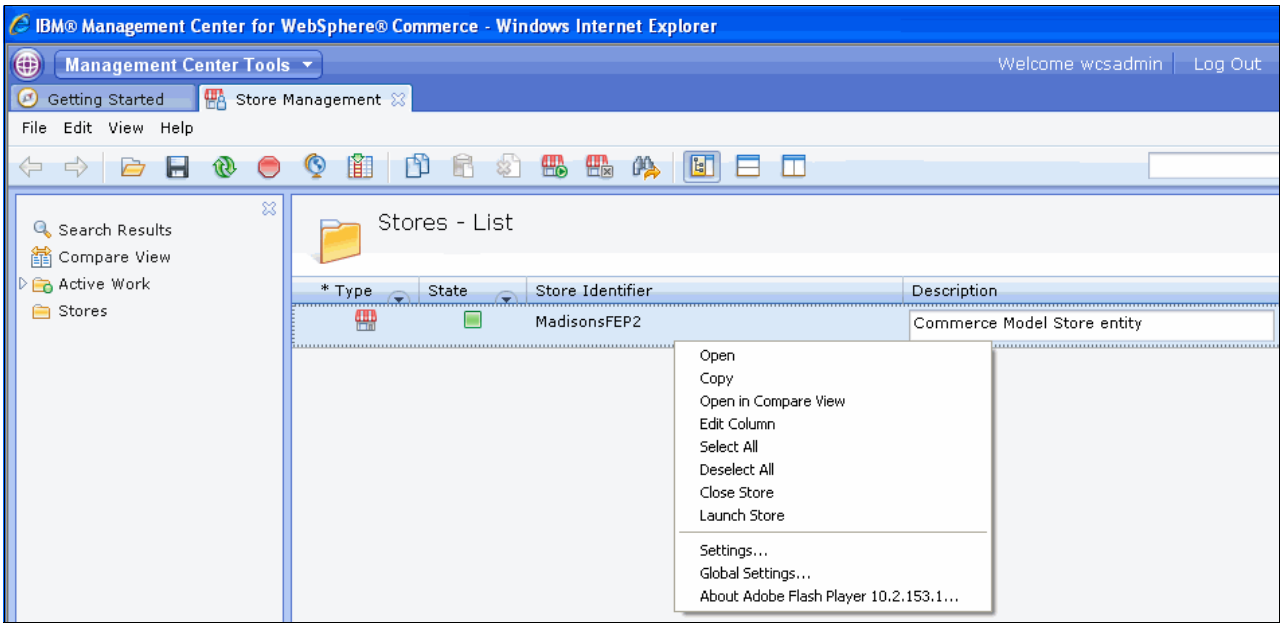


Figure 6-22 List of options in the store menu for the selected store MadisonsFEP2

5. The MadisonsFEP2 store is now opened. Complete these tasks:
   a. With the MadisonsFEP2 store open, select the **Catalog** tab (see Figure 6-23).
   b. Select **Search-based navigation**, which is the last check box in the list.

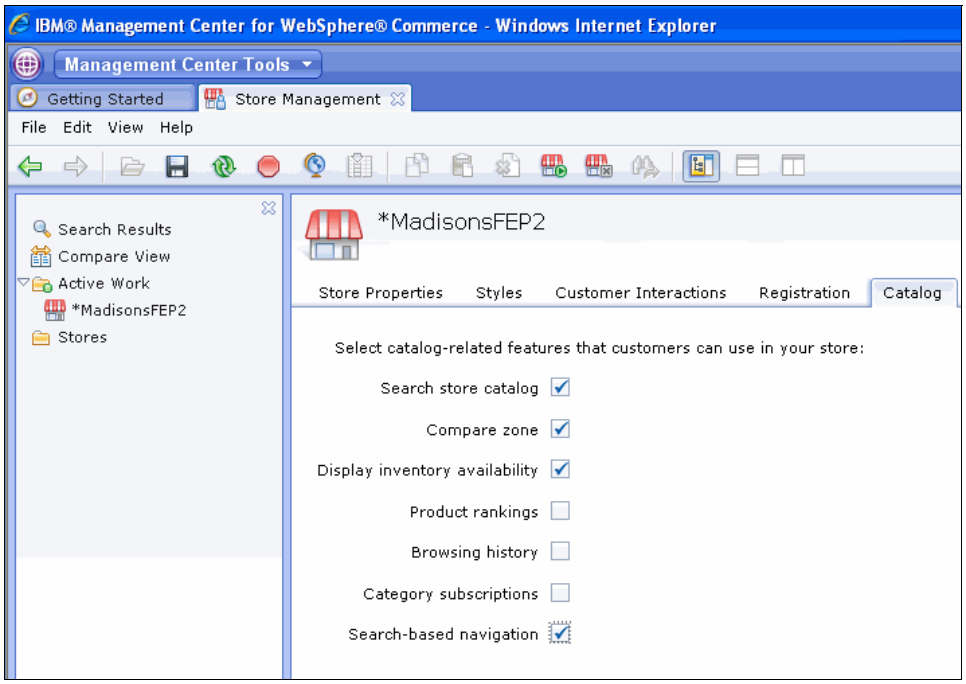


Figure 6-23 Checking Search-based navigation for MadisonsFEP2

6. Click **Save and Close**. See Figure 6-24.

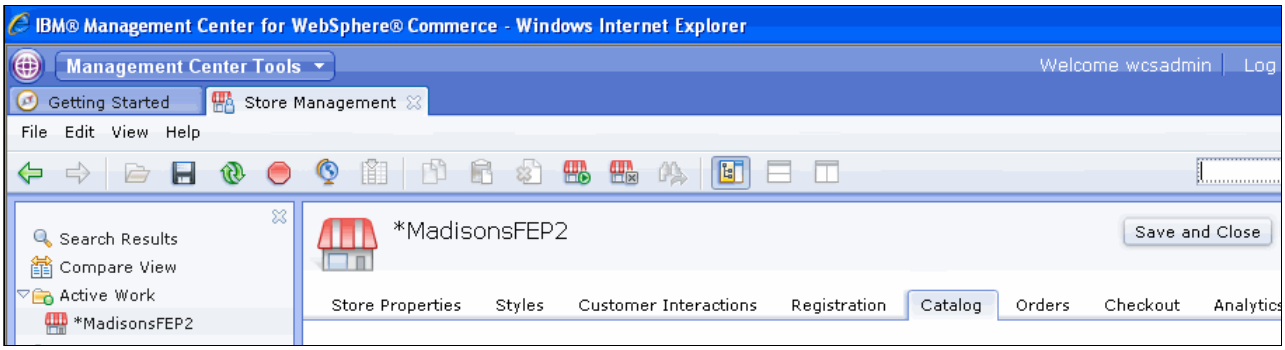


Figure 6-24 Saving and closing the MadisonsFEP2

Your storefront now has search-based navigation enabled.

6.4 Preparing the remote search server

Now, you set up the Solr search on the remote search server and prepare it to work with the WebSphere Commerce machine. First, you install and update WebSphere Application Server on the remote search machine, on which `solr.war` will be deployed and configured to communicate with the Commerce server and DB server. Next, you install and test the DB2 client on the remote search machine. Then, you set up the IBM HTTP Server and Plug-in component on the remote search machine and configure the server for Solr. The last step is to run the command that creates a WebSphere Application Server search profile and all the necessary objects and deploys the `solr.war`.

6.4.1 Preparing the remote search machine procedure

In order to complete the remote search machine preparation, you have to perform all the following activities:

- Installation of WebSphere Application Server with default profile creation and update.
- Installation of the database client.
- Installation of the web server.
- Deployment of the Solr application.
- Configuration of the web server for the Solr application.

6.5 Installing WebSphere Application Server with default profile creation and update

Make sure that the *default* profile is created during the WebSphere Application Server V7.0 installation. You can obtain information about the default profile in the following directory:

`/opt/IBM/WebSphere/AppServer/profiles`

The profile that is named `AppSrv0` is created. You can obtain details about this profile in this file:

`/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/AboutThisProfile.txt`

After completing *all the updating steps* in this section, the WebSphere Application Server V7.0 is updated to IBM Update Installer V7.0.0.15 for WebSphere Application Server V7.0 Fix Pack 15 for Linux 64-bit version.

There are two parts in this section. In the first part, we explain how to update the WebSphere Application Server V7.0 Installer product. And in the second part, we explain how to update the Installer to update the WebSphere Application Server V7.0 product with all necessary packages.

6.5.1 Updating the WebSphere Application Server V7.0 Installer

You must update the WebSphere Application Server V7.0 Installer to the IBM Update Installer V7.0.0.15 for WebSphere Software for Linux 64-bit version. For Help, see this link at the information center:

http://www-01.ibm.com/support/docview.wss?uid=swg24029073

Complete the following steps to install the prerequisites:

1. Install the WebSphere Application Server V7.0 product.
2. Locate this package on the download page:

   `64-bit x86 AMD/Intel (tar.gz)`
3. Download the package using this download link for openSUSE 64bit:

ftp://public.dhe.ibm.com/software/websphere/appserv/support/tools/UpdateInstaller/7.0.x/LinuxAMD64/7.0.0.15-WS-UPDI-LinuxAMD64.tar.gz

Perform the following steps after the download:

1. Extract the downloaded Update Installer package (`*.tar` file) to a temporary directory.
2. Navigate to the **UpdateInstaller** subdirectory.
3. Use the installation wizard to install the Update Installer. To launch the Wizard interface, run the command:

   `./install`
4. After the installation is complete, change to your working directory and run the following update script to install all four update packages for WebSphere Application Server V7.0: AppServer, Plug-ins, IBM HTTP Server (IHS), and software development kit (SDK) (Example 6-3).

Example 6-3 Running the script for Update Installer

```
cd /opt/IBM/WebSphere/UpdateInstaller

./update.sh
```

6.5.2 Updating WebSphere Application Server V7.0

You need to update WebSphere Application Server V7.0 to the 7.0.0.15: WebSphere Application Server V7.0 Fix Pack 15 for Linux version. See the Help link:

http://www-01.ibm.com/support/docview.wss?rs=180&uid=swg24029073

Use the Update Installer to install updates for all of these WebSphere Application Server V7.0 packages:

- 64-bit x86 AMD/Intel® AppServer
- 64-bit x86 AMD/Intel Plug-ins
- 64-bit x86 AMD/Intel IBM HTTP Srvr
- 64-bit x86 AMD/Intel Java SDK

Use this FTP download link list to download update packages for WebSphere Application Server V7.0, Plug-in, IHS, and SDK:

- WebSphere Application Server V7.0:

  ftp://public.dhe.ibm.com/software/websphere/appserv/support/fixpacks/was70/cumulative/cf70015/LinuxX64/7.0.0-WS-WAS-LinuxX64-FP0000015.pak

- Plug-in:

  ftp://public.dhe.ibm.com/software/websphere/appserv/support/fixpacks/was70/cumulative/cf70015/LinuxX64/7.0.0-WS-PLG-LinuxX64-FP0000015.pak

- IBM HTTP Server:

  ftp://public.dhe.ibm.com/software/websphere/appserv/support/fixpacks/was70/cumulative/cf70015/LinuxX64/7.0.0-WS-IHS-LinuxX64-FP0000015.pak

- SDK:

  ftp://public.dhe.ibm.com/software/websphere/appserv/support/fixpacks/was70/cumulative/cf70015/LinuxX64/7.0.0-WS-WASSDK-LinuxX64-FP0000015.pak

This is the complete Linux 64-bit update package list with file names:

- `7.0.0-WS-WAS-LinuxX64-FP0000015.pak`
- `7.0.0-WS-WASSDK-LinuxX64-FP0000015.pak`
- `7.0.0-WS-PLG-LinuxX64-FP0000015.pak`
- `7.0.0-WS-IHS-LinuxX64-FP0000015.pak`

Tip: Do not try to unpack downloaded `*.pak` files. Copy all these files to the `/opt/IBM/WebSPhere/UpdateInstaller/maintenance` folder. Then, during the installation process, point to this location when asked for the Installer maintenance folder.

See Figure 6-25 for the names of the `*.pak` files and the Installer maintenance folder path in the file system.

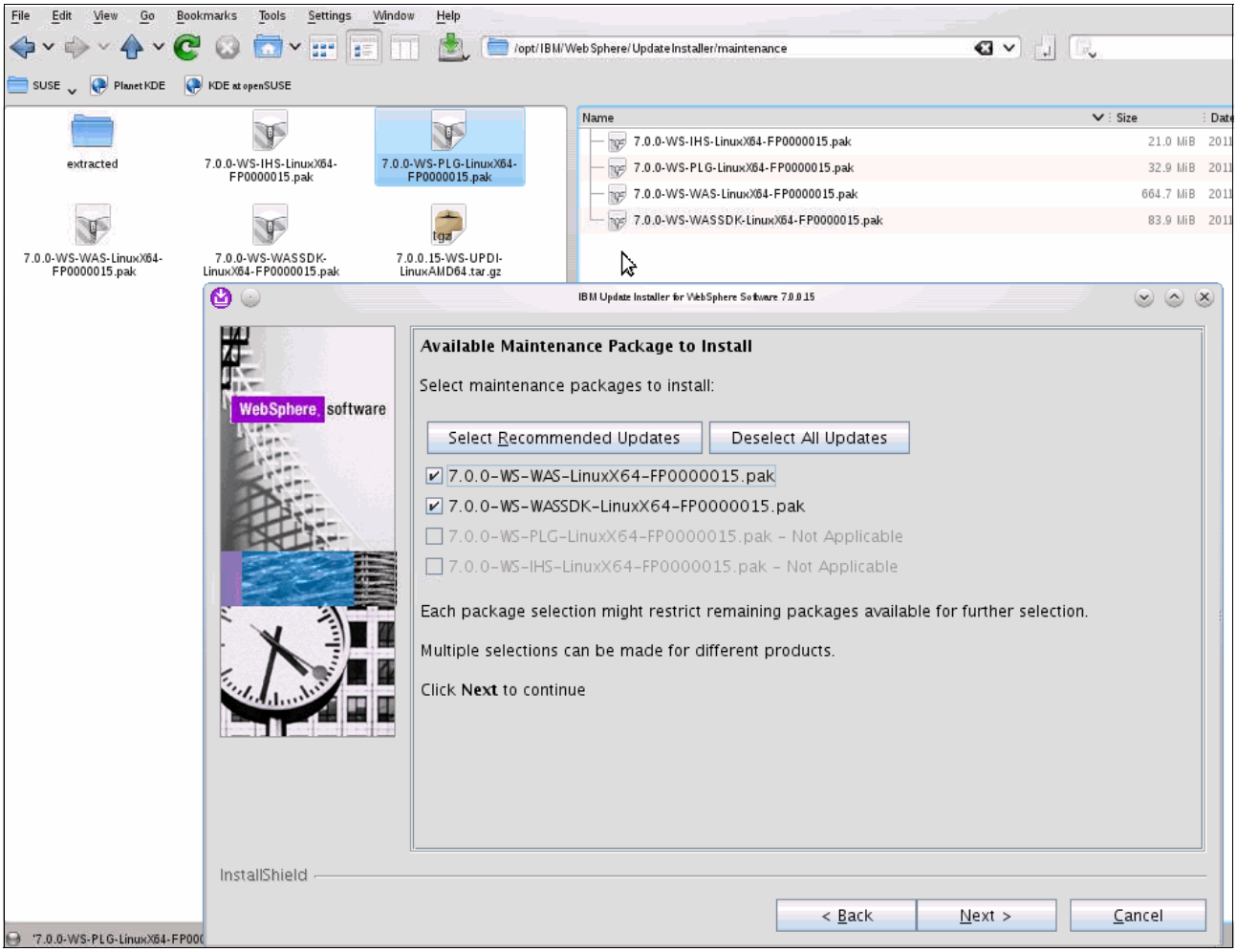


Figure 6-25 Names of package files and Installer maintenance folder path

Perform all the steps in the installation wizard. Verify that your update is successful.

6.6 Installing the database client

WebSphere Commerce V7.0 Feature Pack 2 is set to work with the DB2 Universal Database™ Enterprise Server Edition V9.5.4. for Linux 64-bit version of DB2.

For the search machine, it is necessary to install the database client that matches the WebSphere Commerce machine's database. You need to obtain and install IBM Data Server Client Version 9.5 Fix Pack 4 for Linux 64-bit.

The client installation is straightforward, but there are two important steps that we describe here to make the installation possible on SUSE Linux Enterprise Server 11.4 64-bit. After that, we show you how to connect the installed client to the WebSphere Commerce database server.

Troubleshooting: Make sure that installation path for the client does not contain any blank characters.

Before starting the client installation, make sure that you have installed the necessary libraries. In our example, we need to download and install the `libstdc++33` library, as shown in Figure 6-26.

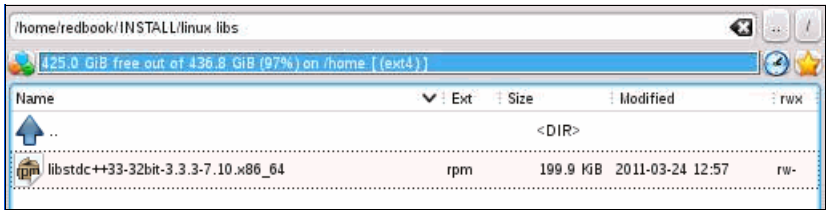


Figure 6-26 Library libstdc++33

Run the command in Figure 6-27 to install the package.

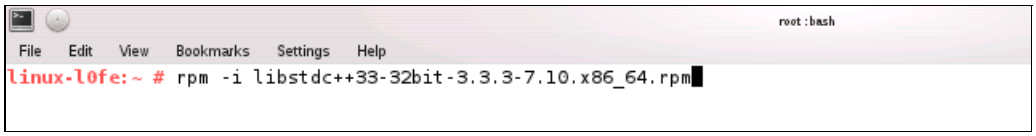


Figure 6-27 Installing libstdc++33

You can use the openSuse administration tool called Yast to locate the `libstd` installed library and to make sure that no files are missing (Figure 6-28).

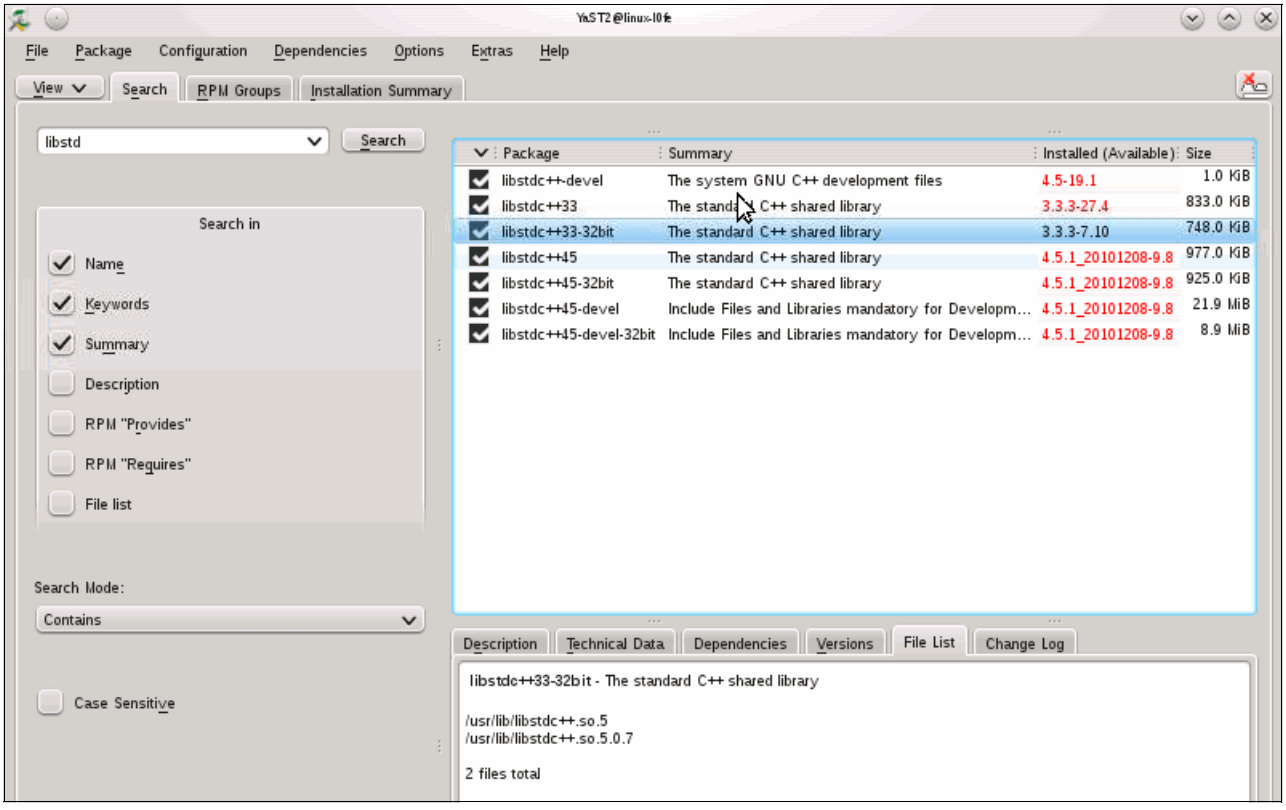


Figure 6-28 Searching for the installed library using the openSUSE tool Yast

After you install your client, try to connect to the remote DB server.

6.6.1 Connecting to the remote DB2 server

On the DB2 server side, you configure the instance for TCP/IP communication and port 50002 is opened to allow remote clients to access data. On the client-side, the DB2 instance is created and the instance name is the same as the user name. You use the installation wizard to set up the DB2 client. The instance owner's information and credentials are provided as step 9 of the installation wizard, and they are shown in Example 6-4 and in Figure 6-29.

Example 6-4 DB2 instance details

```
Instance details:

Username: db2inst1
Password: inst1pas
Home directory: /home/db2inst1
```

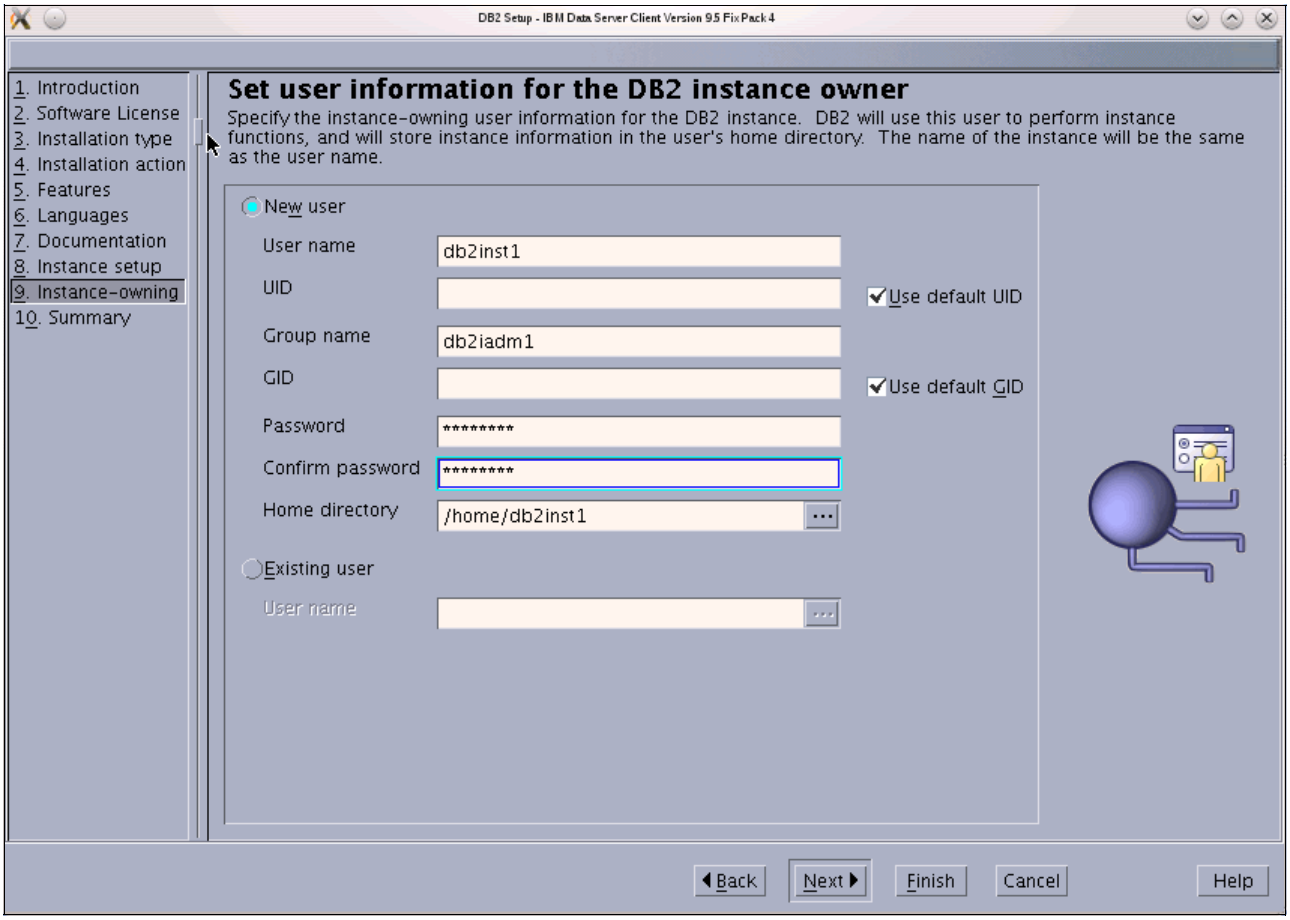


Figure 6-29 DB2 client installation wizard Step 9: Setting up the instance owner's credentials

Perform these steps to connect from the DB2 client on the search machine to the remote Commerce DB2 server:

1. On the remote search machine, where the DB2 client is installed, switch to the `db2inst1` user and issue the command line processor (CLP) invocation command: `db2` (Figure 6-30).

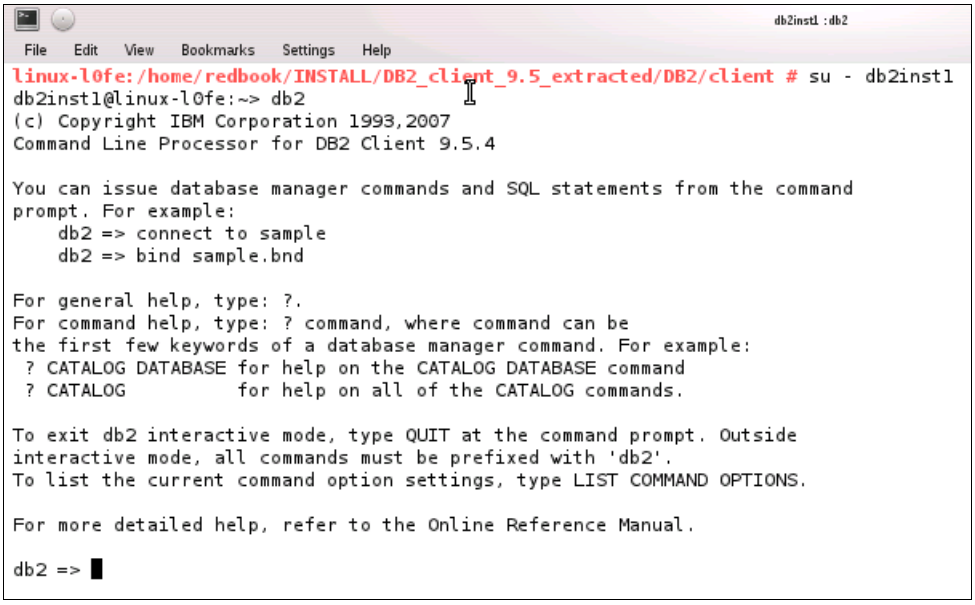


Figure 6-30 Switching to the db2instl DB2 user and invoking the db2 CLP

2. Run the **`catalog tcpip node`** and **`catalog database`** commands using the IP address of the remote DB2 server and the proper port (Figure 6-31). The port number is set during the DB2 server installation.

```
db2 => catalog tcpip node nodeb remote 9.42.171.192 server 50002
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
db2 => catalog database mall as mall at node nodeb
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
db2 =>
```

Figure 6-31 Running the catalog node command and the catalog database command

We describe the commands:

- `CATALOG TCPIP NODE`

  This command adds a TCP/IP node entry to the node directory. The command is run on a client.

- `CATALOG DATABASE`

  This command stores the database location information in the system database directory. The database can be located either on the local workstation or on a remote node.

For the command parameters, Commerce's database name in this example is `mall` and the node name that is used is `nodeb`.

3. Using the DB2 CLP, run the commands as shown in Example 6-5 to list the node and database directory, and check the results in Figure 6-32.

Example 6-5 DB2 commands for listing the database directory and node directory

```
list database directory
list node directory
```

```
db2 => list database directory

 System Database Directory

 Number of entries in the directory = 1

Database 1 entry:

 Database alias                       = MALL
 Database name                        = MALL
 Node name                            = NODEB
 Database release level               = c.00
 Comment                              =
 Directory entry type                 = Remote
 Catalog database partition number    = -1
 Alternate server hostname            =
 Alternate server port number         =

db2 => list node directory

 Node Directory

 Number of entries in the directory = 1

Node 1 entry:

 Node name                      = NODEB
 Comment                        =
 Directory entry type           = LOCAL
 Protocol                       = TCPIP
 Hostname                       = 9.42.171.192
 Service name                   = 50002

db2 =>
```

Figure 6-32 Executing the list database directory and list node directory commands

The verification step ensures that both commands executed successfully.

4. Issue the command in Example 6-6 to connect to the remote database named `mall`. Execute the **`select`** query from a Commerce database table to get the example data (Figure 6-33 on page 145).

Example 6-6 Connecting to the Commerce database and running the select query

```
connect to mall user db2inst1 using inst1pas
select * from userreg
```

In Example 6-6, the name of the Commerce database is `mall`, the database user name is `db2inst1`, and the database password is `inst1pas`.

```
db2 => connect to mall user db2inst1 using inst1pas

   Database Connection Information

 Database server        = DB2/LINUXX8664 9.5.4
 SQL authorization ID   = DB2INST1
 Local database alias   = MALL

db2 => select * from userreg

USERS_ID             STATUS      PLCYACCT_ID LOGONID

              LOGONPASSWORD

RED CHALLENGEQUESTION


                                                                                TIMEOUT              PASSWORD
                       I
                                                                                      PASSWORDCREATION
  OPTCOUNTER
-------------------- ----------- ----------- ----------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------
------------ ----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------
--- ------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------- -------------------- --------
----------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------- ----------------------
- ----------
               -1000           1          -2 wcsadmin

              x'6639475542487274793577624866317 86B583772756E644E44466F6461465A6A6166323232705472634 56F3D2020202020
0202020202020202020202020202020202020202020202020202020202020202020202020202020202020202020202020202020202020202
  0 -


                                                                                                      -1

                                                                                      2011-03-25-02.02.26.000
          1

  1 record(s) selected.

db2 =>
```

Figure 6-33 Connecting to the Commerce database and an example of reading data from the userreg table

The DB2 client on the remote search machine is installed successfully now and ready to use with WebSphere Commerce.

6.7 Installing the web server

In our search example, we use the remote search machine with the Solr application deployed. The Solr application must use a separate web server; therefore, you must install and configure another web server and point it to the Solr profile plug-in file.

On the same search machine where WebSphere Application Server V7.0 with Solr is installed, install IBM HTTP Server Version 7.0.0.15 of the web server. The installation is straightforward. We only show two steps from the installation wizard here. Those two steps affect the overall configuration.

We install the IBM HTTP Server Plug-in for IBM WebSphere Application Server with the following details (Figure 6-34).

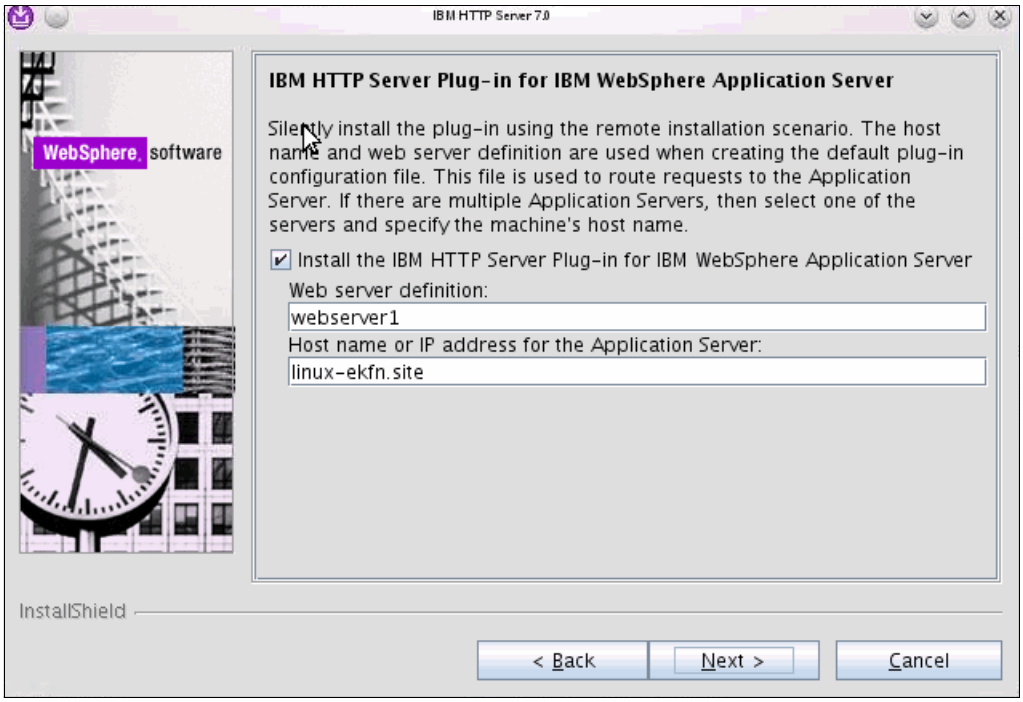


Figure 6-34 Web Server installation wizard showing the plug-in installation details

Figure 6-35 shows the installation summary.

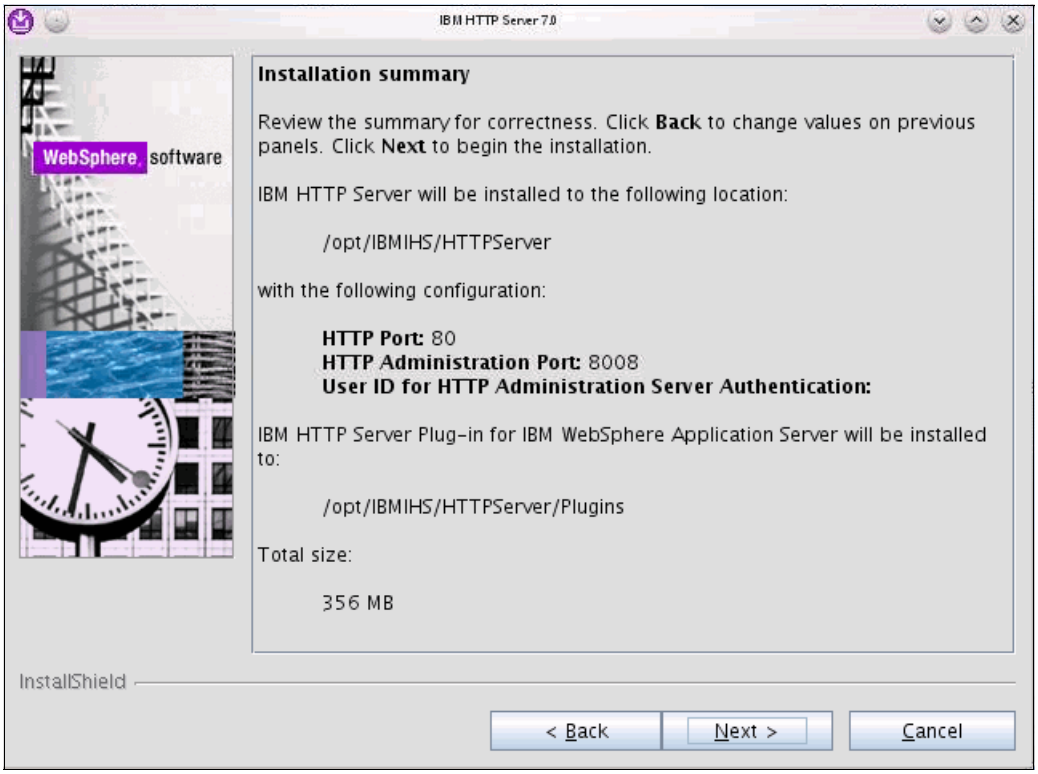


Figure 6-35 Web server installation summary

You can access the web server configuration file here:

`/opt/IBMIHS/HTTPServer/conf/http.conf`

Important: Do not try to run the Solr deployment script (`ws_ant.sh`) before making sure that your `solr-deploy.properties` file contains the proper Web Server Definition Properties. Refer to the Web Server Definition Properties section of the `solr-deploy.properties` file. Locate all web server configuration-related key/value pairs and make sure that they match the proper values in the web server configuration file named `http.conf`.

6.8 Deploying the Solr application

The deployment of the WebSphere Commerce search `solr.war` file is a manual process with provided scripts. We describe this deployment in detail in this chapter.

The WebSphere Commerce FEP2 features enablement generated the relevant property files for search and placed them on the file system of the WebSphere Commerce machine. To deploy the search in advanced mode and on a separate remote search machine, the search files have to be moved from the WebSphere Commerce machine to the remote search machine and updated with the proper, specific values for remote configuration.

On the remote search machine, to deploy `solr.war` to WebSphere Application Server, The deployment script `ws_ant.sh` must perform a set of operations described in the `deploySearch.xml` file. We use the updated values in the `solr-deploy.properties` file during script execution.

The `deploySearch.xml` file is located in `working_dir/search/deploy/deploySearch.xml`.

Example 6-7 lists the tasks in the `deploySearch.xml` file.

Example 6-7 Deployment tasks

```
create “solr home” on the file system.
update solr.war/WEB-INF/web.xml by specifying the solr home string.
create a profile with name <instanceName>_solr.
create authAlias.
create JDBC driver.
create dataSource.
create virtual hosts.
deploy solr.war with application name ‘solr’.
create web server definition.
generate web server plug-in file.
```

For more information, see the information center pages for administering and deploying the WebSphere Commerce search:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdmanagesearch.htm

Troubleshooting: The file location paths, directory paths, script paths, and command examples with parameters that are presented in our examples are unique for our search-specific environment. To understand and use them properly in your environment, see the previous information center link and locate the generic and relative file and directory paths and the full command format for help.

We use these steps in the deployment procedure:

- Copying the search component from the Commerce machine to the remote search machine
- Updating the property values in the `solr-deploy.properties` file
- Running the Solr deployment script

6.8.1 Copying the search component from the WebSphere Commerce machine to the remote search machine

Select the following directory from your WebSphere Commerce machine:

`/opt/IBM/WebSphere/CommerceServer70/components/foundation/subcomponents/search`

Copy it to the working directory, which is `/home/redbooks/INSTALL/` in our example, on your remote search machine:

`/home/redbooks/INSTALL/search`

6.8.2 Updating the property values in the solr-deploy.properties file

Open the properties file:

`/home/redbooks/INSTALL/search/deploy/properties/solr-deploy.properties`

Review the property values that were generated during the feature enablement and update them.

Important: Updates in the `solr-deploy.properties` file are necessary, because properties for WebSphere Application Server V7.0 and IHS contain local Commerce file path values. You must change these file path values to reflect the installation paths of WebSphere Application Server V7.0 and IHS on the remote search server.

In the next example, there are five key/value pairs that need to be updated for this specific remote configuration. They all relate to WebSphere Application Server V7.0 and IHS. All other key/value pairs are unchanged and not displayed in following file extract that is shown in Example 6-8.

Example 6-8 Updated values in the solr-deploy.properties file

```
# System properties for WebSphere Commerce Solr Integration

################################################################################
#
# WebSphere Application Server properties
#
################################################################################
# Fully qualified WebSphere Application Server hostname, e.g., wcsmachine.torolab.ibm.com
#-------------------------------------
WASHostName=linux-530s.site


################################################################################
#
# Web Server Definition properties
#
```

```
################################################################################
# Fully qualified WebServer hostname, e.g., wcsmachine.torolab.ibm.com
#-------------------------------------
webserverHostname=linux-530s.site


# Location of the Web server install, e.g., =/opt/WebSphere/HTTPServer
#-------------------------------------------------------------------------------
webserverInstallLocation=/opt/IBMIHS/HTTPServer

# Webserver configuration file, e.g.,
/opt/WebSphere/CommerceServer70/instances/demo/httpconf/httpd.conf
#-------------------------------------------------------------------------------
webserverConfigFile=/opt/IBMIHS/HTTPServer/conf/httpd.conf

# Location of the plugin install, e.g., /opt/WebSphere/Plugins
#-------------------------------------------------------------------------------
pluginInstallLocation=/opt/IBMIHS/HTTPServer/Plugins
```

6.8.3 Running the Solr deployment script

The following script creates the new WebSphere Application Server profile, all necessary objects, and deploys the `solr.war`.

To deploy the `solr.war`, perform these steps:

1. Navigate to the `bin` directory on WebSphere Application Server:

   ```
   cd /opt/IBM/WebSphere/AppServer/bin
   ```

2. Run the deployment script that is shown in Example 6-9.

   Example 6-9 Running the Solr deployment script

   ```
   ./ws_ant.sh -buildfile /home/redbooks/INSTALL/search/deploy/deploySearch.xml
   -DdbUserPassword=inst1pas -DsolrHome=/home/redbooks/solr_search/
   ```

   In Example 6-9, these parameters have these meanings:

   dbuserpwd The password for the user connecting to the `db2inst1` database.

   solrhome Optional: The location of the Solr Home directory path, which contains the index data of Solr. The value must be an absolute path.

   For this example, we created and used the new directory, `/home/redbooks/solr_search/`.

3. Verify that the build was successful by looking at the log file. See Appendix A, “Log file lookup” on page 251.

Solr deployment results

The solr application installed successfully. Next, we describe the most important details of the Solr deployment:

- Details of Solr Home:

  ```
  name="solr.solr.home" value="/home/redbooks/solr_search/"
  ```

- The new WebSphere Application Server profile `demo_solr` is created. For more information about this profile, look at this website:

  ```
  /opt/IBM/WebSphere/AppServer/profiles/demo_solr/logs/AboutThisProfile.txt
  ```

In Figure 6-36, you can see both profiles on WebSphere Application Server. The first profile, which is called `AppSrv01`, was created during the WebSphere Application Server V7.0 installation.

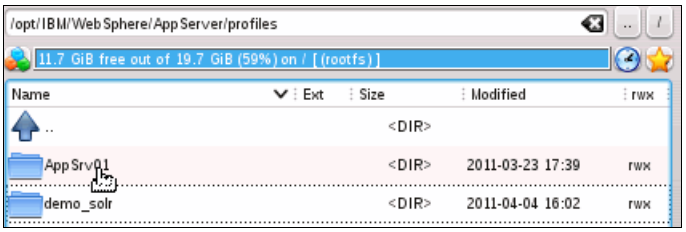


Figure 6-36 WebSphere Application Server V7.0 profiles on the remote search machine

- The server named `solrServer` was ready to use or “opened for e-business” during the `ws_ant.sh` script execution.

Tip: Before you start the initial testing of solr search in your browser, restart **solrServer**.

To restart solrServer and check the server status, perform all steps that are in Example 6-10.

Example 6-10 Restarting solrServer

```
cd /opt/IBM/WebSphere/AppServer/profiles/demo_solr/bin/

./stopServer.sh solrServer
./startServer.sh solrServer
./serverStatus.sh solrServer
```

Use this file for troubleshooting:

`/opt/IBM/WebSphere/AppServer/profiles/demo_solr/logs/solrServer/SystemOut.log`

6.9 Configuring the web server for the Solr application

To configure the web server for the Solr application, you have to update the web server configuration. Port 3737 needs to be opened and you have to point to the Plug-in configuration file path.

Complete the following steps:

1. Open the configuration file:

   `/opt/IBMIHS/HTTPServer/conf/httpd.conf`

2. In the Listen section in the `httpd.conf` configuration file, add a new value for port 3737 (Figure 6-37).

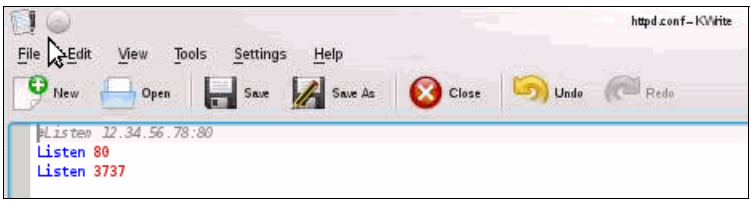


Figure 6-37 Listen section in httpd.conf file

3. In the LoadModule section in the `httpd.conf` configuration file, point to the plug-in configuration file, as shown in Example 6-11.

Example 6-11 Load Module section of httpd.conf file

```
LoadModule was_ap22_module
/opt/IBMIHS/HTTPServer/Plugins/bin/32bits/mod_was_ap22_http.so

/opt/IBM/WebSphere/AppServer/profiles/demo_solr/config/cells/demo_search_cell/n
odes/demo_search_node/servers/solrWebserver/plugin-cfg.xml
```

After changing the `httpd.conf` file, always restart the web server using the steps in Example 6-12.

Example 6-12 Restarting the web server

```
cd /opt/IBMIHS/HTTPServer/bin/

./apachectl.sh stop
./apachectl.sh start
```

6.10 Creating the mappings from the WebSphere Commerce server to the remote server for search term associations

This step is necessary for search term associations to function properly when deploying search remotely. We did not use it in our examples. For more information, refer to the information center:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchbuilddeployremote.htm

6.11 Testing the search deployment

If all previous steps were successful, you can test search deployment.

Follow these steps:

1. Ensure that your search server is up and running.
2. Test your search deployment by navigating to the following URL:

   `http://<hostname>:3737/solr/Default/select?q=*%3A*`

If successful, you receive a response in XML format, which is similar to Example 6-13.

Example 6-13 Testing the search on the Solr server using browser

```
<response>
   <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">9</int>
      <lst name="params">
         <str name="q">*:*</str>
      </lst>
   </lst>
```

```
   <result name="response" numFound="0" start="0" />
</response>
```

6.12 Setting up and deploying the search index

This section describes how to set up the search index and deploy the index structure on the *remote search machine*, after the successful Solr deployment. If you performed all previous advanced deployment steps with no errors, you can run your WebSphere Commerce server and WebSphere Commerce search on separate machines and be able to communicate and perform search operations.

Figure 6-38 shows setting up the WebSphere Commerce search index structure for a specific master catalog in the standard and advanced configuration.

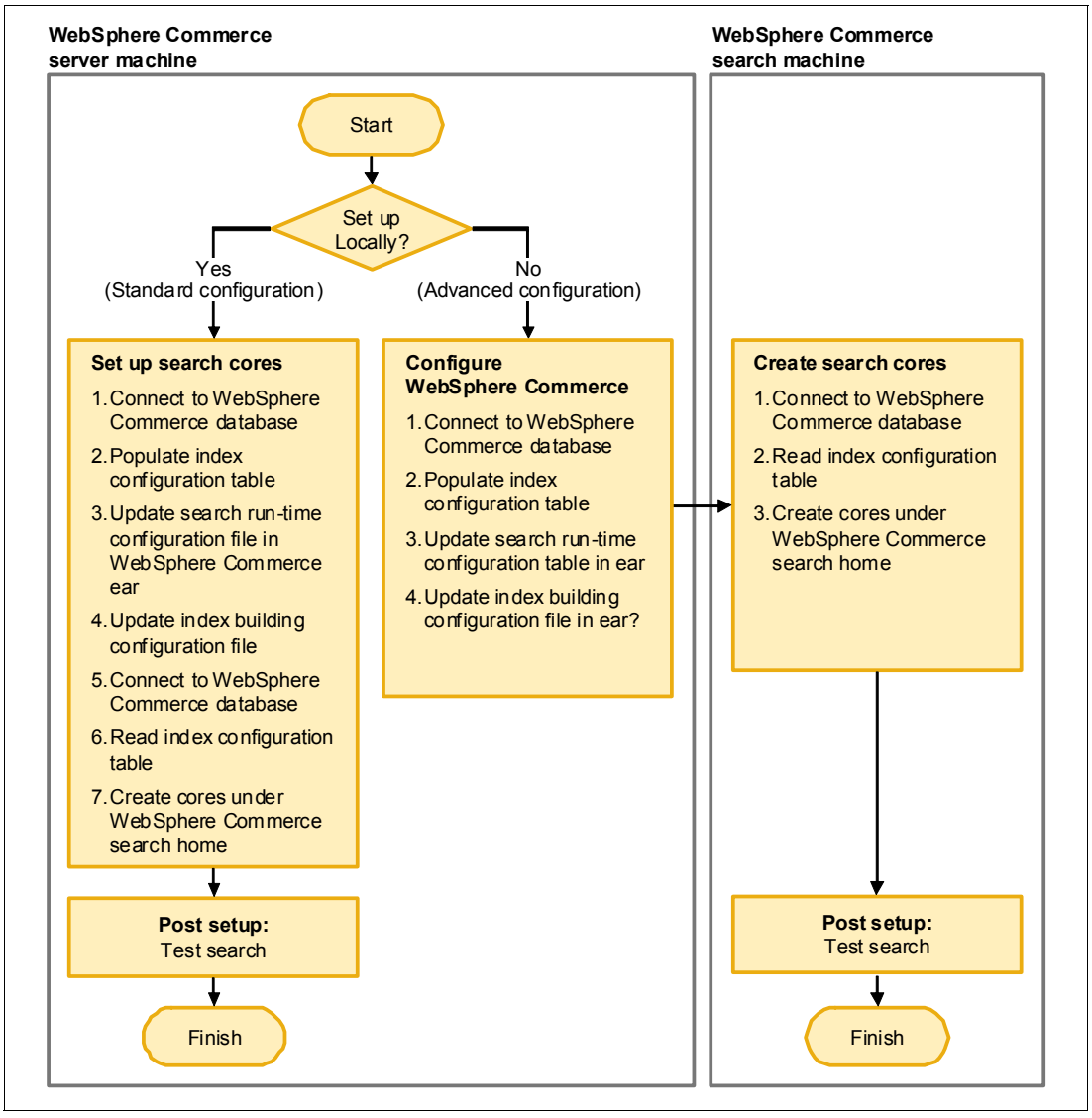


Figure 6-38 Setting up the search index for a master catalog in standard and advanced configuration

6.12.1 Complete indexing procedure description

The indexing procedure requires these steps:

1. Setting up the WebSphere Commerce search index remotely
2. Preprocessing the WebSphere Commerce search index data
3. Building the WebSphere Commerce search index
4. Replicating WebSphere Commerce search

6.13 Setting up the WebSphere Commerce search index remotely

This section describes how to set up the search index structure for a specific master catalog remotely. You can deploy your search index structure on your remote search machine and configure WebSphere Commerce to use the Solr index by running the search index setup utility. This utility ensures that your index is built successfully using your WebSphere Commerce master catalog data.

You run the search index setup utility on the WebSphere Commerce machine first and then on the remote search machine.

6.13.1 Index setup preparation

Make sure that you have finished all the steps in the previous sections for deploying Solr on the remote search machine.

Examine the catalog data model in your Commerce database and make sure that you are familiar with the search-related tables, languages, stores, and catalogs after the enablement of FEP2.

If you do not know the details about your catalog, run the following SQL:

```
select * from catalog where IDENTIFIER='YOUR_STORE_IDENTIFIER'
```

In our example (Example 6-14), we use the following master catalog ID.

Example 6-14 SQL query to obtain the master catalog ID based on the store identifier

```
select catalog_id from catalog where IDENTIFIER='MadisonsFEP2'

Result:
catalog_id
----------
10001
```

Run this SQL script to find the initial data in the search-related tables, as shown in Example 6-15.

Example 6-15 SQL scripts for search-related tables

```
select * from srchattr
select * from srchattrprop
select * from srchconf
```

Tip: Review all the values in the configuration database table SRCHCONF after performing the indexing procedure. You can obtain the detailed configuration of the index in the column SRCHCONF.CONFIG.

6.13.2 Preparing the WebSphere Commerce machine

On the WebSphere Commerce machine, perform the following steps:

1. Log on as a WebSphere Commerce non-root user.
2. Navigate to the following directory:

   ```
   cd /opt/IBM/WebSphere/CommerceServer70/components/foundation/subcomponents/search/bin
   ```

3. Run the search index setup utility, as shown in Example 6-16.

   Example 6-16 Running the search index setup utility

   ```
   ./setupSearchIndex.sh -instance demo -action configWCforSolr -masterCatalogId
   10001 -dbuser db2inst1 -dbuserpwd inst1pas -searchServerName linux-530s
   -searchServerPort 3737 -searchServiceContextRoot /solr
   ```

   In Example 6-16, the parameters have these definitions:

   instance_name The name of the WebSphere Commerce instance with which you are working.

   masterCatalogId The ID of the master catalog.

   dbuser The name of the DB2 user connecting to the database.

   dbuserpwd The password for the DB2 user connecting to the database.

   searchServerName The remote search server host name, where Solr is deployed.

   searchServerPort The search server port. The default value is 3737.

   searchServiceContextRoot The search service context root. For the Solr-related action, such as configSolrCores or configWCforSolr, the default value is `/solr`.

   indextype Optional: Search engine index to set up. The default value is `CatalogEntry`.

4. Ensure that the utility runs successfully and that the search index setup successfully completes without any errors.
5. If the setup does not complete successfully and displays warnings or errors, use the `wc-search-index-setup.log` file (Figure 6-39) for help with debugging.

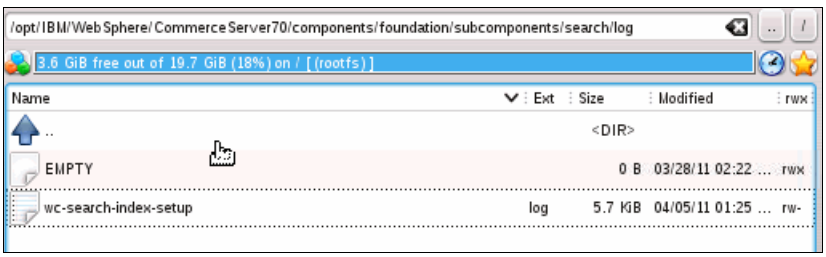


Figure 6-39 Location of wc-search-index-setup.log file

The log file contains the details about all the executed steps (Example 6-17 on page 155).

Example 6-17 setupSearchIndex.sh utility execution result

```
================================================================
WebSphere Commerce Search Index Setup Utility
================================================================

Utility started at 2011-04-05 01:22:57
Configuration initialization started at 2011-04-05 01:22:57
Configuration initialization finished at 2011-04-05 01:22:58
Database initialization started at 2011-04-05 01:22:58
Database initialization finished at 2011-04-05 01:22:58
Initialization completed in 1.578 seconds.
Build and execute process job started at 2011-04-05 01:22:58


************************************************************
Started configuring WebSphere Commerce for Solr cores
************************************************************
Master Catalog Id: 10001
Index Type: CatalogEntry
Languages: [en_US]
************************************************************

--- Started configuring pre-process. ---
--- Finished configuring pre-process. ---

--- Started configuring classic attribute. ---
--- Finished configuring classic attribute. ---

--- Started configuring attribute dictionary. ---
--- Finished configuring attribute dictionary. ---

--- Started configuring delta update. ---
--- Finished configuring delta update. ---

--- Started configuring wc-search.xml. ---
Started exporting the wc-search.xml from EAR. It will take a few minutes.
profileName=demo
registry=/opt/IBM/WebSphere/AppServer/properties/profileRegistry.xml
profileHome=/opt/IBM/WebSphere/AppServer/profiles/demo
Finished exporting the wc-search.xml from EAR.

Started updating the wc-search.xml.
Finished updating the wc-search.xml.

Started importing the updated wc-search.xml to EAR. It will take a few minutes.
profileName=demo
registry=/opt/IBM/WebSphere/AppServer/properties/profileRegistry.xml
profileHome=/opt/IBM/WebSphere/AppServer/profiles/demo

Finished importing the updated wc-search.xml to EAR.
--- Finished configuring wc-search.xml. ---


************************************************************
Finished configuring WebSphere Commerce for Solr cores
```

```
************************************************************


Build and execute process job finished at 2011-04-05 01:25:14

Program exiting with exit code: 0.
Search index setup successfully completed without errors.

Utility stopped at 2011-04-05 01:25:14
Total running time is 137.361 seconds.
```

Procedure results

The search index setup is complete, the file `wc-search.xml` is updated, and one record was inserted into the database table SRCHCONF.

You can access the file `wc-search.xml` in this folder:

`/opt/IBM/WebSphere/AppServer/profiles/demo/installedApps/WC_demo_cell/WC_demo.ear/xml/config/com.ibm.commerce.catalog-fep`

In Example 6-18, you can see an extract from the `wc-search.xml` file with the advanced (remote) configuration details and core details.

Example 6-18 File wc-search.xml extract with configuration and core details

```
  ==================== Server configurations ====================
   -->
    <!--
       This setting uses the Apache Commons HTTP Client to connect to Solr.
   -->
    <_config:server name="AdvancedConfiguration">
        <_config:common-http URL="http://linux-530s:3737/solr/"
            allowCompression="true" connectionTimeout="5000"
            defaultMaxConnectionsPerHost="100" followRedirects="false"
            maxRetries="2" maxTotalConnections="100" soTimeout="2000"/>
    </_config:server>

...

 <!--
       ========================================
       Search indexes to be used for references
       ========================================

       <_config:index deltaUpdate="true" fullBuild="false"
         name="CatalogEntry"
object="com.ibm.commerce.catalog.facade.server.services.search.metadata.solr.SolrCatalogNav
igationViewImpl"/>
     <_config:index name="UnstructuredContent"
object="com.ibm.commerce.catalog.facade.server.services.search.metadata.solr.SolrCatalogNav
igationViewImpl"/>

   <_config:cores>

      <_config:core catalog="10001" indexName="CatalogEntry"
           language="en_US" name="MC_10001_CatalogEntry_en_US"
           path="/MC_10001/en_US/CatalogEntry"
           serverName="AdvancedConfiguration" synonym="/conf/synonyms.txt"/>
```

```
        <_config:core catalog="10001" indexName="UnstructuredContent"
            language="en_US"
            name="MC_10001_CatalogEntry_Unstructured_en_US"
            path="/MC_10001/en_US/CatalogEntry/unstructured"
            serverName="AdvancedConfiguration" synonym="/conf/synonyms.txt"/>

    </_config:cores>
    -->
```

The configuration table for the WebSphere Commerce search integration was changed during the script `setupSearchIndex.sh execution`. One record was inserted into the database table SRCHCONF, as shown in Example 6-19.

Example 6-19 Search configuration in database

```
select * from srchconf

INDEXTYPEINDEXSCOPELANGUAGESCONFIGOPTCOUNTER
--------------------------------------------------------------------
CatalogEntry10001-1
IndexScopeTag=0,SearchServerPort=3737,SearchServerName=linux-530s,PreProcessConfigDirect
ory=/opt/IBM/WebSphere/CommerceServer70/instances/demo/search/pre-processConfig/MC_10001
/DB2   0

  1 record(s) selected.
```

6.13.3 Preparing the remote search machine

On the remote search machine, perform all the steps in the following procedure.

Preparation steps

Before running the search index setup utility, you need to review all the values in the `solr-deploy.properties` file and you must use the proper libraries in order for this step to succeed:

1. Open the `solr-deploy.properties` properties file:

   `/home/redbooks/INSTALL/search/deploy/properties/solr-deploy.properties`

   Review the property values that were generated during the feature enablement and update them to match your remote search machine configuration.

2. The database initialization uses the native library named `db2jcct2`. To prevent a failure in loading this library during the execution of the index setup utility, run this command:

   ```
   export LD_LIBRARY_PATH=/home/db2inst1/sqllib/lib64:$LD_LIBRARY_PATH
   ```

Perform the following steps:

1. Log on as a non-root user.
2. Navigate to the `bin` directory:

   ```
   cd /home/redbooks/INSTALL/search/bin
   ```

3. Run the search index setup utility from the `working_dir/search` directory, as shown in Example 6-20 on page 158.

Example 6-20 Running the search index setup utility

```
./setupSearchIndex.sh -instance demo -setupMode remote -action configSolrCores
-masterCatalogId 10001 -dbuser db2inst1 -dbuserpwd inst1pas -wcServerName
linux-fnai -wcServerPort 80 -wasHome /opt/IBM/WebSphere/AppServer -solrhome
/home/redbooks/solr_search/
```

In Example 6-20, the parameters have these meanings:

instance_name	The name of the WebSphere Commerce instance with which you are working.
setupMode remote	Required for WebSphere Commerce Developer to be set up as remote on WebSphere Commerce. Optional for run time to indicate explicitly that a remote setup is used.
masterCatalogId	The ID of the master catalog.
dbuser	The name of the DB2 user connecting to the database.
dbuserpwd	The password for the DB2 user connecting to the database.
dbname	Optional: The DB2 database name to be connected.
dbport	Optional: The DB2 database port to be connected.
dbserver	Optional: The DB2 database server host location.
dbtype	Optional: The database type, for example, cloudscape, DB2, or Oracle.
wcServerName	The WebSphere Commerce Web Server host name.
wcServerPort	The port of your WebSphere Commerce Web Server.
WAS_home	The WebSphere Application Server home directory.
solrhome	Optional: The location of the Solr Home directory path, which contains the index data of Solr. The value must be an absolute path. The default value is: `working_dir/search/instance_name/search/solr/home` For this example, we created and used this directory for Solr Home: `/home/redbooks/solr_search/`
indextype	Optional: Search engine index to set up. The default value is CatalogEntry.

4. Ensure that the utility runs successfully and that the search index setup successfully completes without warnings or errors. If setup does not complete or it shows warnings or errors, use the `wc-search-index-setup.log` file for help with debugging. See Example 6-21 for details.

Example 6-21 Script setupSearchIndex.sh utility execution result

```
Using remote setup mode.
Using WAS Java. [Recommend]

================================================================
WebSphere Commerce Search Index Setup Utility
================================================================

Utility started at 2011-04-05 05:09:57
Configuration initialization started at 2011-04-05 05:09:57
```

```
Configuration initialization finished at 2011-04-05 05:09:58
Database initialization started at 2011-04-05 05:09:58
Database initialization finished at 2011-04-05 05:09:58
Initialization completed in 0.54 seconds.
Build and execute process job started at 2011-04-05 05:09:58

************************************************************
Started setup for Solr cores
************************************************************
Master Catalog Id: 10001
Index Type: CatalogEntry
Languages: [en_US]
************************************************************

--- Started checking directories for setup Solr cores. ---
--- Finished checking directories for setup Solr cores. ---

************************************************************
The following languages passed directory validation:
- en_US

The following Solr cores will set up:
- MC_10001_CatalogEntry_en_US
- MC_10001_CatalogEntry_Unstructured_en_US

************************************************************

--- Started copying Solr core files. ---
--- Finished copying Solr core files. ---

--- Started configuring DIH (wc-data-config.xml). ---
--- Finished configuring DIH (wc-data-config.xml). ---

--- Started registering Solr core in solr.xml. ---
--- Finished registering Solr core in solr.xml. ---

Build and execute process job finished at 2011-04-05 05:09:58

Program exiting with exit code: 0.
Search index setup successfully completed without errors.

Utility stopped at 2011-04-05 05:09:58
Total running time is 0.887 seconds.
```

5. Notice the change in the file system on the remote search machine, as shown in Figure 6-40 on page 160.

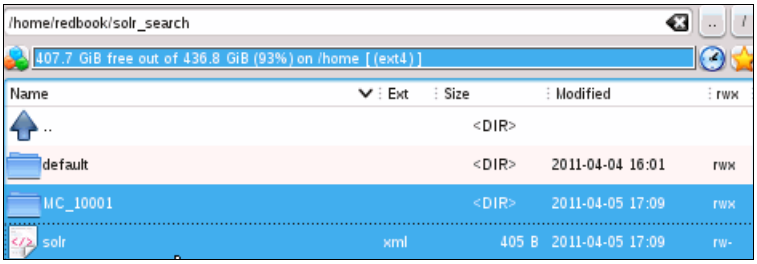


Figure 6-40 Solr home with MC_10001 folder and solr.xml file created on remote search machine

6. Figure 6-41 shows the content of the `MC_10001` folder.

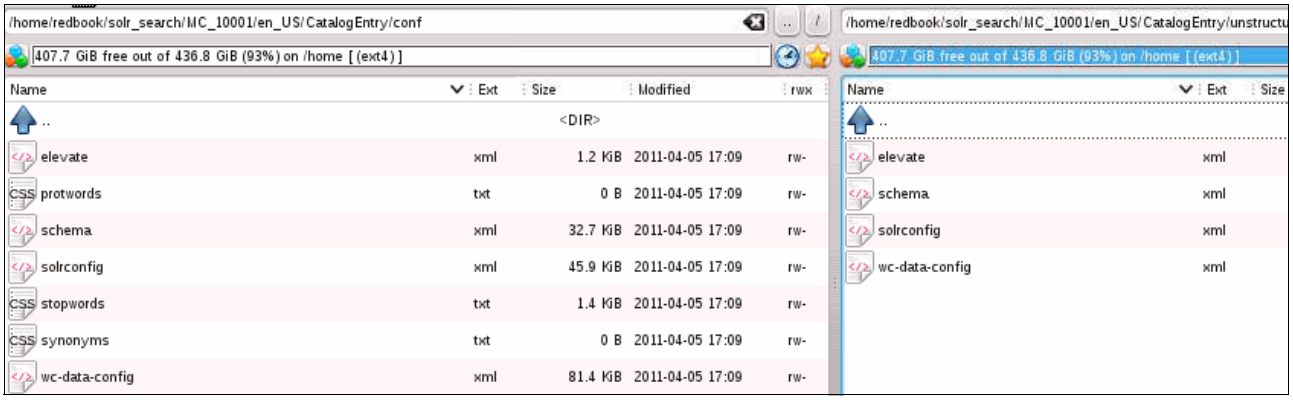


Figure 6-41 Content of MC_10001 folder

Two core instance tags were added to the `/home/redbooks/solr_search/solr.xml` file, under the default instance, as shown in Figure 6-42.

```
<?xml version="1.0" encoding="UTF-8"?>
<solr persistent="true" sharedLib="lib">
    <cores adminPath="/admin/cores">
        <core instanceDir="default" name="Default"/>
        <core instanceDir="MC_10001/en_US/CatalogEntry/" name="MC_10001_CatalogEntry_en_US"/>
        <core instanceDir="MC_10001/en_US/CatalogEntry/unstructured/" name="MC_10001_CatalogEntry_Unstructured_en_US"/>
    </cores>
</solr>
```

Figure 6-42 The solr.xml file with two new cores

After setting up the search index structure for a specific master catalog, you must preprocess the search index data, which we describe in the next section.

6.14 Preprocessing the WebSphere Commerce search index data

You must preprocess the search index data to prepare your WebSphere Commerce data for indexing.

The preprocess utility extracts and flattens WebSphere Commerce data and then outputs the data into a set of temporary tables inside the WebSphere Commerce database. The data in the temporary tables is from the base schema and is then used by the index building utility to populate the data into Solr indexes using Solr's Data Import Handler (DIH).

Complete the following steps:

1. Ensure that you have completed setting up the WebSphere Commerce search index structure for a specific master catalog as described in the previous chapter.
2. On the WebSphere Commerce machine, log on as a non-root user.
3. Navigate to the following directory:

   ```
   cd /opt/IBM/WebSphere/CommerceServer70/bin>
   ```

4. Run the preprocessing utility, as shown in Example 6-22.

   Example 6-22 Running preprocessing utility

   ```
   ./di-preprocess.sh
   /opt/IBM/WebSphere/CommerceServer70/instances/demo/search/pre-processConfig/MC_
   10001/DB2 -instance demo -dbuser db2inst1 -dbuserpwd inst1pas -fullbuild true
   ```

   In Example 6-22, the following parameters have these definitions:

   full-path Required: The full path location of the load order configuration file, for example:

   `WC_installdir/instances/demo/search/pre-processConfig/MC_10001/DB2`

   MC_10001 represents masterCatalogID 10001.

   instance_name The name of the WebSphere Commerce instance.

   dbuser The name of the DB2 user connecting to the database.

   dbuserpwd The password for the DB2 user connecting to the database.

   fullbuild The flag indicating whether it is a full index build. The accepted values are either `true` or `false`. The default value is `true`.

   localename The locale to index. The default value is `All`. The accepted values are either `All` or one of the following values:

   `en_US, fr_FR, de_DE, it_IT, es_ES, pt_BR, zh_CN, zh_TW, ko_KR, ja_JP, ru_RU, ro_RO, pl_PL,` or `ar_EG`

5. Ensure that the preprocessing utility runs successfully without errors or warnings.

6.14.1 Result

The preprocess utility extracted WebSphere Commerce data and put the data into a set of temporary tables inside the WebSphere Commerce database.

You can obtain the path to the log file `wc-dataimport-preprocess.log` in Appendix A, “Log file lookup” on page 251. Example 6-23 shows an extract of the log file with the preprocessing script execution details.

Example 6-23 Extract of preprocess utility log

```
Apr 5, 2011 5:38:29 AM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain logStartDate
INFO: Data import pre-processing started:Tue Apr 05 05:38:29 EDT 2011

Apr 5, 2011 5:38:30 AM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain main
INFO: Data import pre-processing initialization completed in 0.539 seconds.
```

```
Apr 5, 2011 5:38:31 AM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
...
Apr 5, 2011 5:38:32 AM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain
processDataConfig
INFO: Data import pre-processing completed in 1.707 seconds for table TI_CATENTRY_0.

...
Apr 5, 2011 5:38:40 AM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor
process(DataProcessingConfig, Connection, boolean, String)
INFO: Sales catalog processing complete.
Apr 5, 2011 5:38:40 AM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor
process(DataProcessingConfig, Connection, boolean, String)
...
INFO:
======================================================================
Catalog hierarchy search index data pre-processor
======================================================================
Batch size: 10000
Total catalog entries that have had their hierarchy determined: 812
======================================================================

...
Apr 5, 2011 5:38:40 AM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain
processDataConfig
INFO: Data import pre-processing completed in 1.089 seconds for table TI_APGROUP_0.
...

INFO:
Program exiting with exit code: 0.
Data import pre-processing completed successfully with no errors.
...
```

For a complete list of temporary tables and their definitions, follow this link:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.developer.doc/concepts/csdsearchindextemptables.htm

After preprocessing the search index data, you must build the search index, which we describe in the next section.

6.15 Building the WebSphere Commerce search index

You can build the WebSphere Commerce search index by using the index building utility.

The index building utility is a wrapping utility that updates the information in the master index using the Data Import Handler (DIH) service to build the index, either partially through delta index updates or completely through full-index builds. When there are multiple indexes, for example, each language using its own separate index, the index is built multiple times.

Preparation steps

Ensure that you have finished all the steps from the previous section, 6.14, “Preprocessing the WebSphere Commerce search index data” on page 160. Ensure that you have published `wc.ear` on the Commerce server and ensure that your Commerce server is started.

Complete the following steps:

1. On the WebSphere Commerce machine, log on as a non-root user.
2. Navigate to the bin directory:

   ```
   cd /opt/IBM/WebSphere/CommerceServer70/bin>
   ```

3. Run the index building utility, as shown in Example 6-24.

   Example 6-24 Running the index building utility

   ```
   ./di-buildindex.sh -instance demo -masterCatalogId 10001 -dbuser db2inst1
   -dbuserpwd inst1pas
   ```

   In Example 6-24, the parameters have the following definitions:

masterCatalogId	The ID of the master catalog.
dbuser	The name of the DB2 user connecting to the database.
dbuserpwd	The password for the DB2 user connecting to the database.
fullbuild	The flag indicating whether this build is a full-index build. The accepted values are either `true` or `false`. The default value is `true`.
statusInterval	The interval in milliseconds that the utility uses to check the index building status. The default value is 10,000 milliseconds.
localename	The locale to index. The accepted values are either `All` or one of the following values: `en_US`, `fr_FR`, `de_DE`, `it_IT`, `es_ES`, `pt_BR`, `zh_CN`, `zh_TW`, `ko_KR`, `ja_JP`, `ru_RU`, `ro_RO`, `pl_PL`, or `ar_EG`

4. Ensure that the index build is successful.
5. Result: The utility reports the status of the indexing progress based on the `statusInterval` parameter and by default prints every 10 seconds the number of documents indexed in each index, how long the utility has run, and the current indexing status. After the utility completes, it reports how many index builds failed. In Example 6-25, you can see the details from the build index utility log showing how Commerce, DB2, and search servers communicate.

Example 6-25 Build index utility log

```
Apr 5, 2011 5:53:19 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain logStartDate(Date)
INFO: Data import process started:Tue Apr 05 05:53:19 EDT 2011
Apr 5, 2011 5:53:19 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
buildSearchConfigFromDBJ2SE
INFO: Build search configuration from database: jdbc:db2://linux-fnai.site:50002/mall.
Apr 5, 2011 5:53:19 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain executeDIH
INFO: The Solr server is linux-530s, the port is 3737.
Apr 5, 2011 5:53:20 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain executeDIH
INFO: Solr server core MC_10001_CatalogEntry_Unstructured_en_US status is initialized.
Apr 5, 2011 5:53:20 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain executeDIH
INFO: Solr server core MC_10001_CatalogEntry_en_US status is initialized.
```

```
Apr 5, 2011 5:53:31 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain fullDataImport

INFO: Solr server core MC_10001_CatalogEntry_Unstructured_en_US status is
{responseHeader={status=0,QTime=1},initArgs={defaults={config=wc-data-config.xml}},command=
status,status=idle,importResponse=,statusMessages={Total Requests made to DataSource=1,
Total Rows Fetched=8, Total Documents Skipped=0, Full Dump Started=2011-04-05 17:53:23,
=Indexing completed. Added/Updated: 4 documents. Deleted 0 documents., Committed=2011-04-05
17:53:25, Optimized=2011-04-05 17:53:25, Total Documents Processed=4, Time taken
=0:0:1.802},WARNING=This response format is experimental.  It is likely to change in the
future.}.

Apr 5, 2011 5:53:31 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain fullDataImport
INFO:

Apr 5, 2011 5:53:41 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain fullDataImport

INFO: Solr server core MC_10001_CatalogEntry_en_US status is
{responseHeader={status=0,QTime=1},initArgs={defaults={config=wc-data-config.xml}},command=
status,status=idle,importResponse=,statusMessages={Total Requests made to DataSource=3,
Total Rows Fetched=782, Total Documents Skipped=0, Full Dump Started=2011-04-05 17:53:34,
=Indexing completed. Added/Updated: 778 documents. Deleted 0 documents.,
Committed=2011-04-05 17:53:35, Optimized=2011-04-05 17:53:35, Total Documents
Processed=778, Time taken =0:0:1.406},WARNING=This response format is experimental.  It is
likely to change in the future.}.

Apr 5, 2011 5:53:41 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain fullDataImport
INFO:

Apr 5, 2011 5:53:41 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain logExitCode
INFO:

------------------------------------------------

Apr 5, 2011 5:53:41 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain logExitCode(int)
INFO:
Program exiting with exit code: 0.
Data import process completed successfully with no errors.

Apr 5, 2011 5:53:41 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain logExitCode
INFO:
------------------------------------------------


Apr 5, 2011 5:53:41 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain logEndDateAndTime
INFO: Data import process ended:Tue Apr 05 05:53:41 EDT 2011
Apr 5, 2011 5:53:41 AM
com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain logEndDateAndTime
INFO: Data import process completed in 22.455 seconds.
```

6. On Figure 6-43 on page 165 and Figure 6-44 on page 165, you can see changes in the file system on the remote search machine in the Solr Home directory, after the build index script completes.

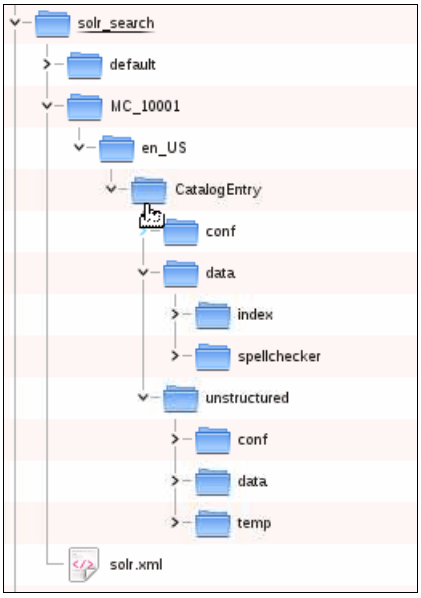


Figure 6-43 Solr home and folder structure after building index

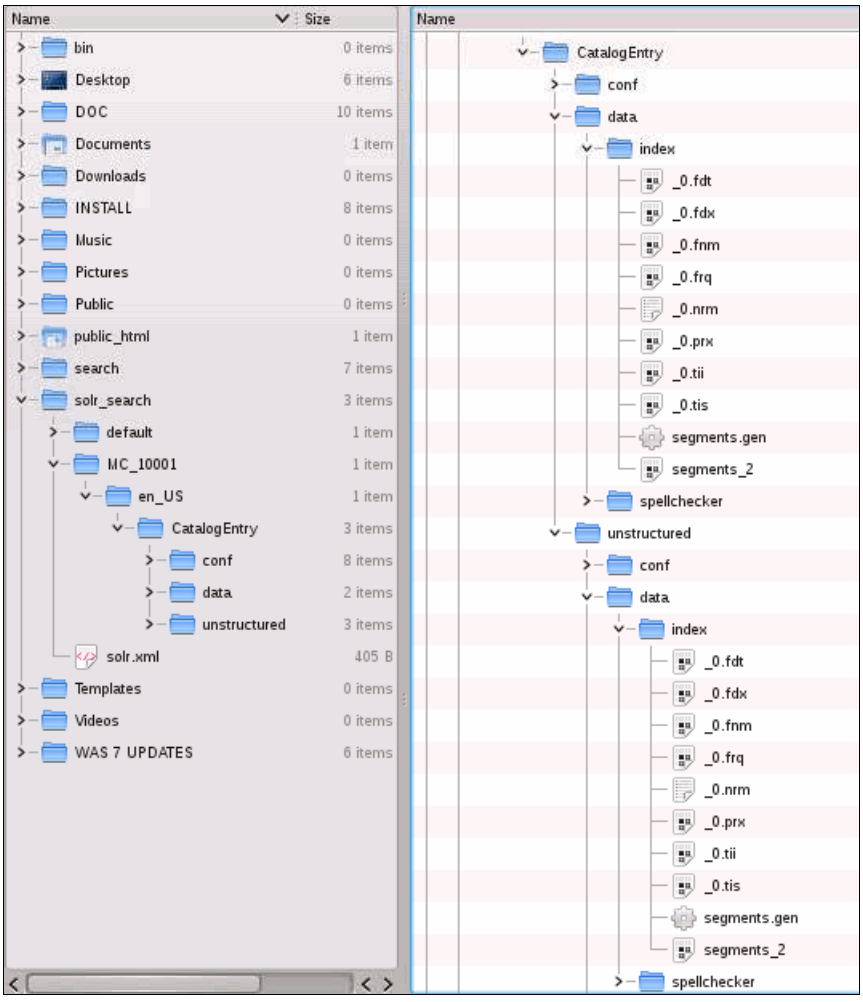


Figure 6-44 Solr home and created files after building index

After building the search index, you can replicate the WebSphere Commerce search, which we describe in the next section.

6.16 Replicating the WebSphere Commerce search

Replication is the process of synchronizing a local copy of an index with changes that have been made on a remote search machine.

The search index data (index files) that were created in the previous section during the building an index process can be replicated from one search machine (master) to another search machine (slave) to improve reliability, fault-tolerance, and performance.

Replicating WebSphere Commerce search maintains your search index data in more than one location, and involves copying designated changes for one location to another and synchronizing the data in both locations. For more information, go to a Solr replication presentation using this link:

http://wiki.apache.org/solr/SolrReplication

6.16.1 Preparation

Ensure that you have completed building the WebSphere Commerce search index that was described in the previous section. After this preparation, perform all the steps on the master (source) search machine first and then on the slave (target) search machine described next, considering the Solr Home setting.

The Solr Home on the source machine is set to `home/redbooks/solr_search/`. The Solr Home on the target machine is set to `/home/redbooks/search/demo/search/solr`.

6.16.2 Configuring the search master machine

Perform the following steps:

1. On the search master machine, go to the solr profile directory:

   ```
   cd /opt/IBM/WebSphere/AppServer/profiles/demo_solr/bin
   ```

2. Stop the solr search server:

   ```
   ./stopServer.sh solrServer
   ```

3. Open the following file:

   ```
   home/redbooks/solr_search/MC_10001/en_US/CatalogEntry/conf/solrconfig.xml
   ```

   In this file, replication is set up for United States English (`en_US`). If you want to replicate for another language, change `en_US` to the locale of your choice.

4. In the `solrconfig.xml` file, locate the following snippet that is shown in Example 6-26.

Example 6-26 Replication snippet in solrconfig.xml as an XML comment

```
<!--  Please refer to http://wiki.apache.org/solr/SolrReplication for details
on configuring replication -->
<!--  remove the <lst name="master"> section if this is just a slave -->
<!--  remove  the <lst name="slave"> section if this is just a master -->
<!--
<requestHandler name="/replication" class="solr.ReplicationHandler" >
    <lst name="master">
```

```
      <str name="replicateAfter">commit</str>
      <str name="replicateAfter">startup</str>
      <str name="confFiles">schema.xml,stopwords.txt</str>
    </lst>
    <lst name="slave">
      <str name="masterUrl">http://localhost:8983/solr/replication</str>
      <str name="pollInterval">00:00:60</str>
    </lst>
</requestHandler>
-->
```

5. Edit the snippet from Example 6-26 on page 166, providing the details of the `master` section and deleting all other sections in this tag, as shown in Example 6-27.

Example 6-27 Replication snippet in solrconfig.xml for master

```
<requestHandler name="/replication" class="solr.ReplicationHandler">
   <lst name="master">
   <str name="replicateAfter">commit</str>
   <str name="replicateAfter">startup</str>
   <str name="confFiles">schema.xml,stopwords.txt</str>
   </lst>
</requestHandler>
```

6. Save your changes and close the file.
7. Restart your search server:

```
./startServer.sh solrServer
```

Ensure that all steps were successful.

6.16.3 Configuring the search slave machine

Perform these steps on your search slave machine:

1. Stop your search server:

```
./stopServer.sh solrServer
```

2. Open the following file:

```
/home/redbooks/search/demo/search/solr/MC_10001/en_US/CatalogEntry/conf/solrconfig.xml
```

3. Similar to the previous steps for the master search, find the `requestHandler` tag, remove the master details, and provide only the slave details, as shown in Example 6-28.

Example 6-28 Replication snippet in solrconfig.xml for slave

```
<requestHandler name="/replication" class="solr.ReplicationHandler">
   <lst name="slave">
   <str
name="masterUrl">http://linux-53Os:3737/solr/MC_10001_CatalogEntry_en_US/replication</str>
   <str name="pollInterval">00:00:60</str>
   </lst>
</requestHandler>
```

4. Save your changes and close the file.

5. Browse through the file system on slave machine and ensure that the folder named `data` does *not* exist at this moment on this folder path:

   `/home/redbooks/search/demo/search/solr/MC_10001/en_US/CatalogEntry/data`

Folder data will be created and populated with the replicated data after you hit this URL on the search slave machine:

`http://linux-ekfn:3737/solr/MC_10001_CatalogEntry_en_US/select?q=*:*`

If you do not encounter any errors or warnings during this procedure, you can see the result of the replication by querying the slave search machine, as shown in Figure 6-45 and Figure 6-46 on page 169.

Figure 6-45 Querying the slave search machine

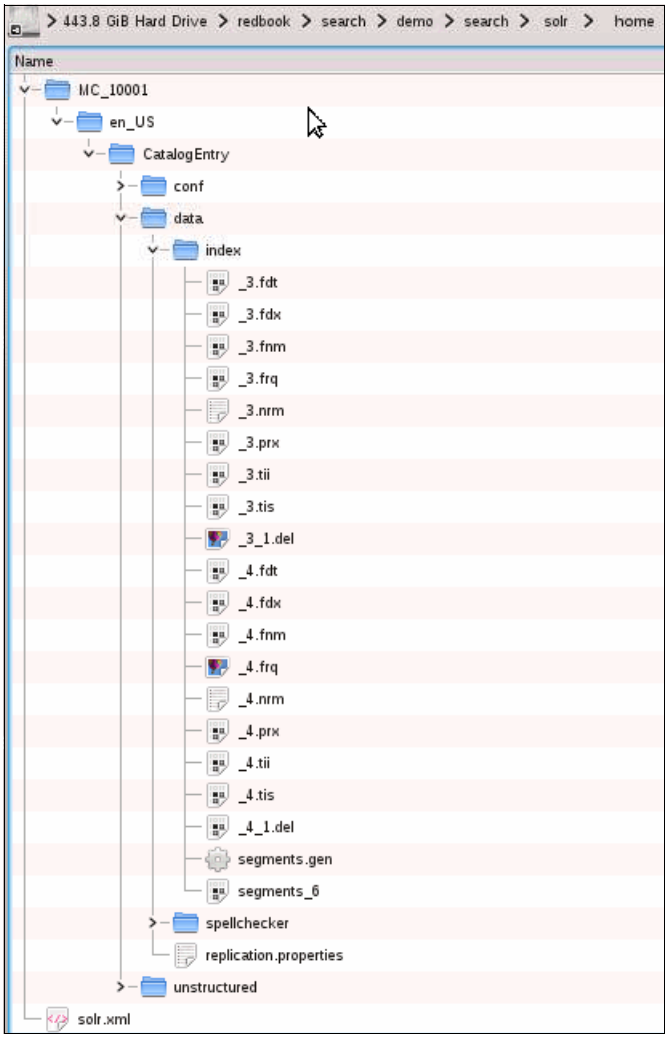


Figure 6-46 Pointing queries from Commerce to a separate search server

After you have set up the replication to push indexes to dedicated search machines, you need to set up WebSphere Commerce to direct the queries to this dedicated search machine. Currently, queries are still being sent to the dedicated indexing machine.

You can set up WebSphere Commerce to direct the queries to the dedicated search machine by modifying the `wc-search.xml`, which is located here:

`WC_ear_dir``/xml/config/com.ibm.commerce.catalog-fep/wc-search.xm`

You must not directly modify this file, but instead create a copy. Then, place and modify this copy in a new directory, `com.ibm.commerce.catalog-ext`. Refer to the information center document:

`http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdwcsearchcom.htm`

Two sections need to be changed in the `wc-search.xml`: the Server section and the Core Section.

In the wc-search.xml, locate the Server section (see Example 6-29).

Example 6-29 wc-search.xml snippet server section

```
<_config:server name="AdvancedConfiguration">
        <_config:common-http URL="http://linux-530s:3737/solr/"
            allowCompression="true" connectionTimeout="5000"
            defaultMaxConnectionsPerHost="100" followRedirects="false"
            maxRetries="2" maxTotalConnections="100" soTimeout="2000"/>
    </_config:server>
```

This section is the HTTP configuration for sending queries to the dedicated index (Master) machine. Add the following tag directly after this section (see Example 6-30).

Example 6-30 wc-search.xml Server section addition

```
<_config:server name="DedicatedSearch">
        <_config:common-http URL="http://linux-ekfn:3737/solr/"
            allowCompression="true" connectionTimeout="5000"
            defaultMaxConnectionsPerHost="100" followRedirects="false"
            maxRetries="2" maxTotalConnections="100" soTimeout="2000"/>
    </_config:server>
```

The second line Example 6-30 is the URL for reaching the dedicated search (slave) machine. The first line is the naming identifier. Now, locate the following lines in the Core section in the wc-search.xml (see Example 6-31).

Example 6-31 wc-search.xml Core section

```
<_config:cores>
        <_config:core catalog="10001" indexName="CatalogEntry"
            language="en_US" name="MC_10001_CatalogEntry_en_US"
            path="/MC_10001/en_US/CatalogEntry"
            serverName="AdvancedConfiguration" synonym="/conf/synonyms.txt"/>
        <_config:core catalog="10001" indexName="UnstructuredContent"
            language="en_US"
            name="MC_10001_CatalogEntry_Unstructured_en_US"
            path="/MC_10001/en_US/CatalogEntry/unstructured"
            serverName="AdvancedConfiguration" synonym="/conf/synonyms.txt"/>
    </_config:cores>
```

Change the bolded "**AdvancedConfiguration**" values to "**DedicatedSearch**". Save the changes. The EAR must be updated with these changes.You can use a single file update in the WebSphere Application Server console to update the EAR with these changes. For instructions to perform a single file update, refer to this information center page:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tdpdeploying_j2ee_assets_single.htm

After the file is updated, run a quick query on the storefront. Then, check the systemOut.log file in both the dedicated indexing machine, and the dedicated search machine. The search query is now visible in the dedicated search machine.

6.17 Advanced search configuration and heterogeneous deployment scenarios

We provide a brief overview of four heterogeneous deployment scenarios and then we describe the last example scenario in detail. All scenarios are based on important decisions that can make a big impact on your search solution and its real-time performance, reliability, and maintenance.

The advanced search configuration gives you the flexibility and scalability for your search solution. You can have multiple WebSphere Commerce server nodes and WebSphere Commerce search nodes with the workload balancer managing traffic to the servers.

With this advanced configuration, your search servers do not need to have all the search data for each catalog. Each master catalog has at least two cores: one core for structured content and one core for unstructured content. These cores can be split among the servers. This section discusses core distribution among your servers, considerations when distributing cores, and possible configurations.

Never distribute the cores among the search Java virtual machines (JVMs) in a way that will minimize the load to any one server. For example, assume that you have three stores, each with its own master catalog, and two search servers. One master catalog, master catalog A, gets 50% of all traffic. One possible configuration is to have master catalog A on its own search server, and place the other two catalog cores on the other search server.

There are other considerations, for example, such as failover. Putting a core on only one search machine means that if that search machine goes down, so does the search for that core. From a failover perspective, it is best to have several machines responsible for a core.

Another possible factor to consider is how often your various catalogs are updated. If one catalog is rarely updated, and another catalog is updated frequently, it might make sense to split these cores up. You can then greatly decrease the synchronization checks on the servers containing the core with the infrequent updates.

Additionally, each language has its own core. For example, if you have a master catalog in three languages, you have six cores for that master catalog (one structured core and one unstructured core for each language). It is also possible to split up languages onto separate search JVMs.

Maintaining separate index cores for each language offers these major benefits:

- Clean schema design

  The text search schema is language-dependent. For example, separate languages contain separate analyzers. Certain languages contain spell checking or stemming, while others do not.

- Runtime query performance

  Shoppers typically only search using one language. The query then only needs to run against one index.

- Easier maintenance for language dictionaries

Each language contains separate synonyms, stop words, and rules.

In the next four charts, we provide four possible scenarios to show you several possible configurations: Figure 6-47 on page 172, Figure 6-48 on page 172, Figure 6-49 on page 172,

and Figure 6-50 on page 173. Both Solr servers can be set up remotely, but you can always set up one Solr server locally and a second Solr server remotely.

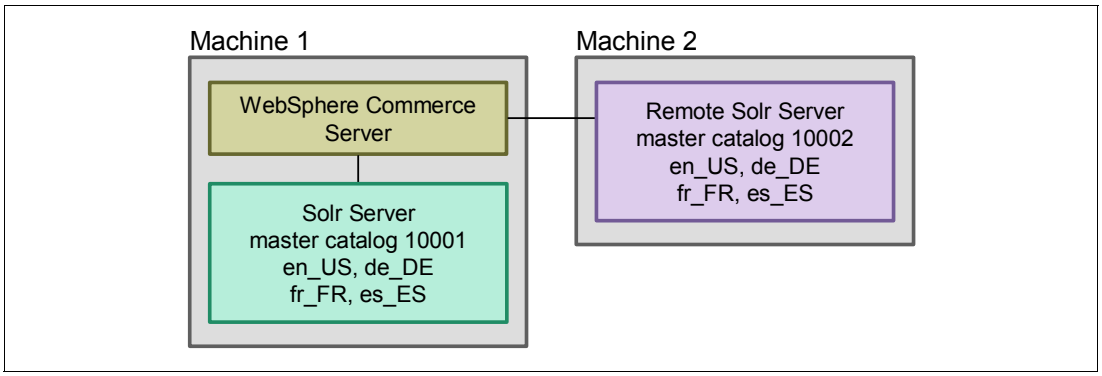


Figure 6-47 Separate master catalogs on separate Solr servers (one local and one remote)

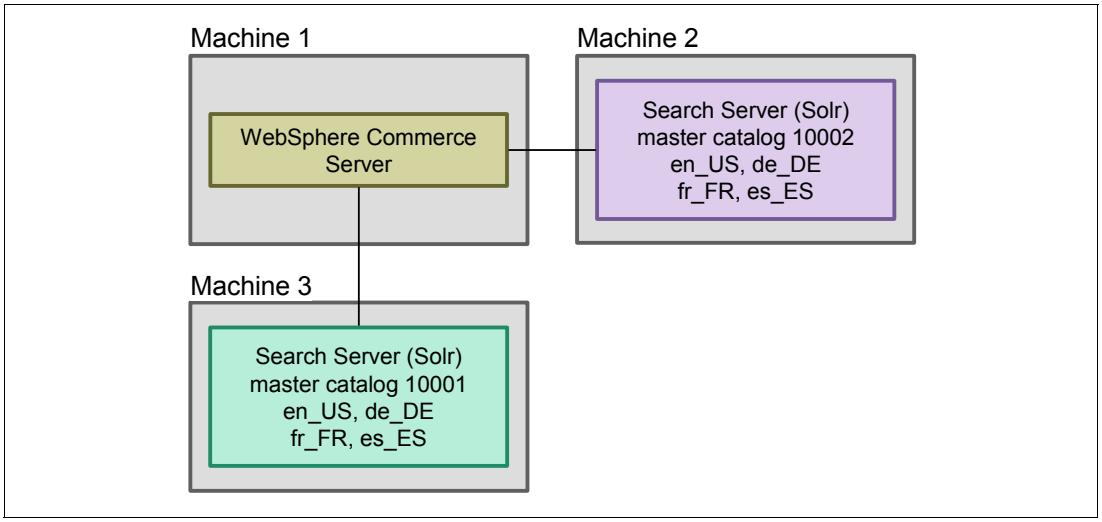


Figure 6-48 Separate master catalogs on separate Solr servers with both Solr servers set up remotely

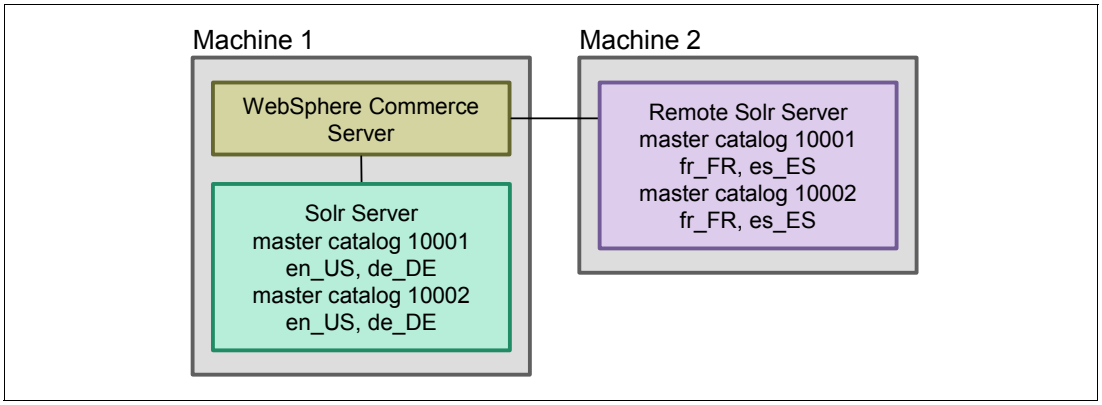


Figure 6-49 Separate languages on separate Solr servers: one local server and one remote Solr server

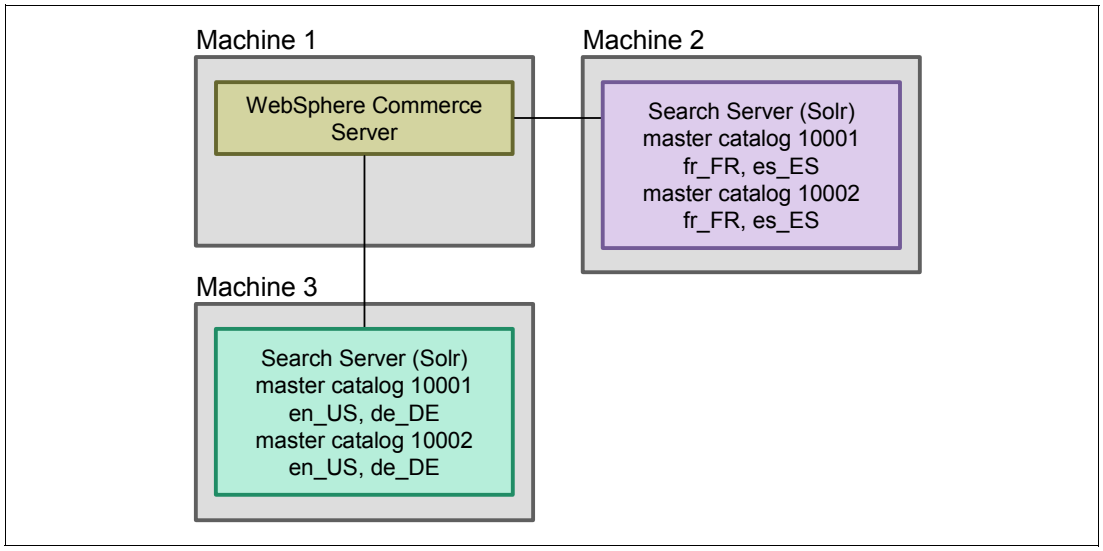


Figure 6-50 Separate languages on separate Solr servers with both Solr servers set up remotely

6.17.1 Setting up language cores on separate Solr servers

This example builds on the scenario setup that we have used throughout this chapter. As a brief overview, currently, there are two search servers set up remotely from the Commerce server: a dedicated indexing (master) machine and a dedicated search (slave) machine. This example adds a language to the store, and a new search server. We configure this search server to serve all queries for this new language.

Figure 6-51 on page 174 shows the overall configuration.

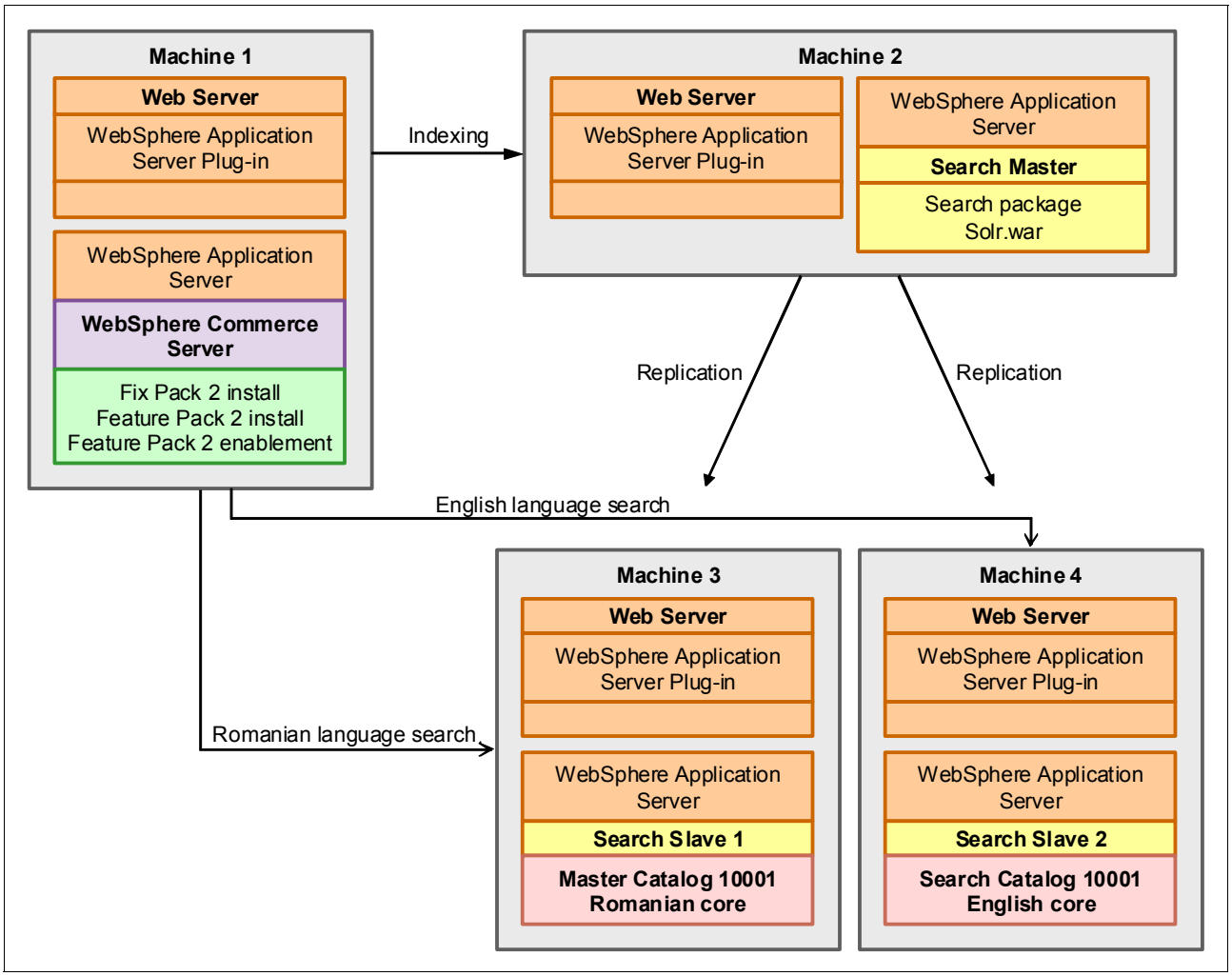


Figure 6-51 Overall configuration with separate languages on separate Solr servers

Adding a language to the storefront

The Madisons sample store, by default, includes multiple languages that can be displayed on the storefront. The languages to display are configurable through the Management Center.

To add a new language, navigate to the **Store Management** tab in the Management Center. Locate the Supported languages section, and add a new language, as shown in Figure 6-52.

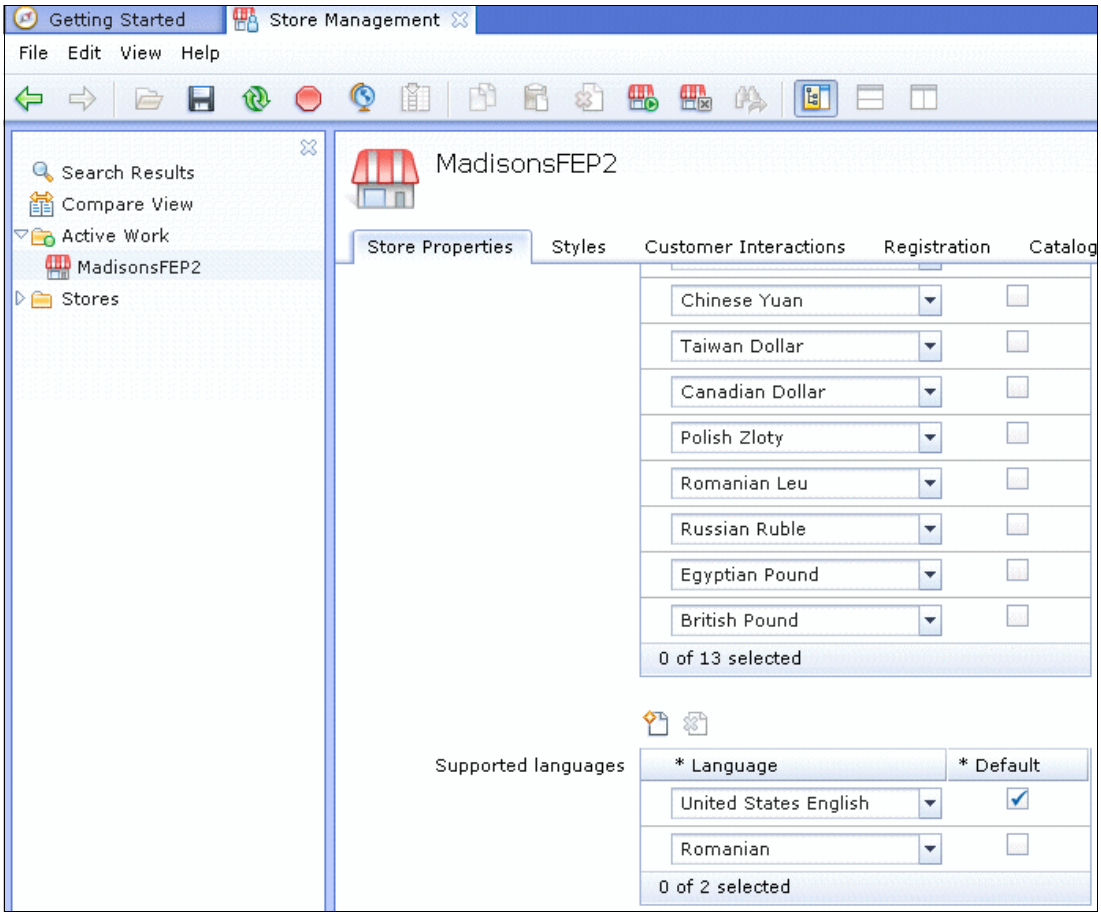


Figure 6-52 Store Properties tab

Save your changes.

Now, Romanian is displayable in the storefront. Or it might be, except that search-based navigation has already been selected for this store. Search-based navigation requires search indexes to be built for your store. As of yet, no cores exist for the Romanian language.

Adding search cores for the Romanian language

When `setupSearchIndex.sh` was run, it was only run for the English language. Now that Romanian has been added, setupSearchIndex needs to be run again. It needs to be run on the WebSphere Commerce machine, as well as the *dedicated indexing machine*. `SetupSearchIndex.sh` needs to be run on the new search machine as well; alternatively, you can copy the search directory structure from the *dedicated indexing machine* to the new search machine.

You can obtain the instructions for running setupSearchIndex on the WebSphere Commerce machine in 6.13.2, “Preparing the WebSphere Commerce machine” on page 154. After `setupSearchIndex.sh` completes, you see that the new language (-21) has been added to the srchconf table (see Example 6-32 on page 176).

Example 6-32 Results of Select languages from srchconf

```
LANGUAGES
--------------
-1,-21
```

Important: Remember, your changes to `wc-search.xml` in the `com.ibm.commerce.catalog-fep` directory will be lost. Therefore, queries will be directed to the dedicated indexing machine.

Now, run `SetupSearchIndex.sh` on the dedicated indexing machine, as described in 6.13.3, “Preparing the remote search machine” on page 157. After `SetupSearchIndex.sh` completes successfully, you see the new ro_RO directory tree under `/home/redbooks/solr_search/MC_10001/`. Make sure to restart the Solr server after running `setupSearchIndex.sh`. If you do not, an error occurs during `di-buildindex.sh`.

Next, run `di-preprocess.sh` and `di-buildindex.sh` by following the instructions in 6.14, “Preprocessing the WebSphere Commerce search index data” on page 160 and 6.15, “Building the WebSphere Commerce search index” on page 162. You can build indexes for both languages, or you can specify only the new language to be built by adding the parameter `-localename ro_RO`.

At this point, the storefront works for the Romanian language, although all queries are being served directly on the dedicated search machine.

Next, configure a new Solr server. Adding additional search servers is a relatively easy, quick procedure. For this example, the new server is set up on an entirely separate machine, by following the instructions that are in 6.5, “Installing WebSphere Application Server with default profile creation and update” on page 137 through 6.8, “Deploying the Solr application” on page 147. Then, run `setupSearchIndex.sh` again by following the instructions that are in 6.13.3, “Preparing the remote search machine” on page 157.

After the new search server is up and running, configure replication from the dedicated indexer to the new dedicated search machine. The dedicated indexing machine is already set up for replication, so only the new search server needs to be set up, by following the instructions that are in 6.16.3, “Configuring the search slave machine” on page 167.

The modified `solrConfig.xml` section for the new search server looks like Example 6-33.

Example 6-33 solrConfig.xml section

```
   <requestHandler name="/replication" class="solr.ReplicationHandler">
      <lst name="slave">
      <str
   name="masterUrl">http://linux-530s:3737/solr/MC_10001_CatalogEntry_ro_RO/replic
   ation</str>
      <str name="pollInterval">1:00:00</str>
      </lst>
</requestHandler>
```

The final step is to set up the `wc-search.xml` to get WebSphere Commerce to direct queries to the proper machines. The Server and Core sections need to be updated.

The newly modified server section looks like Example 6-34 on page 177.

Example 6-34 Server section in wc-search.xml

```
<_config:server name="AdvancedConfiguration">
        <_config:common-http URL="http://linux-530s:3737/solr/"
            allowCompression="true" connectionTimeout="5000"
            defaultMaxConnectionsPerHost="100" followRedirects="false"
            maxRetries="2" maxTotalConnections="100" soTimeout="2000"/>
    </_config:server>
    <_config:server name="DedicatedSearch">
        <_config:common-http URL="http://linux-ekfn:3737/solr/"
            allowCompression="true" connectionTimeout="5000"
            defaultMaxConnectionsPerHost="100" followRedirects="false"
            maxRetries="2" maxTotalConnections="100" soTimeout="2000"/>
    </_config:server>

   <_config:server name="DedicatedRomanian">
        <_config:common-http URL="http://linux-l0fe:3737/solr/"
            allowCompression="true" connectionTimeout="5000"
            defaultMaxConnectionsPerHost="100" followRedirects="false"
            maxRetries="2" maxTotalConnections="100" soTimeout="2000"/>
    </_config:server>
```

The host name of the new Solr server is `http://linux-l0fe:3737/solr/`. `DedicatedRomanian` is the new Solr server identifier.

The newly configured core section looks like Example 6-35.

Example 6-35 wc-search.xml core section

```
<_config:cores>
        <_config:core catalog="10001" indexName="CatalogEntry"
            language="en_US" name="MC_10001_CatalogEntry_en_US"
            path="/MC_10001/en_US/CatalogEntry"
            serverName="DedicatedSearch" synonym="/conf/synonyms.txt"/>
        <_config:core catalog="10001" indexName="UnstructuredContent"
            language="en_US"
            name="MC_10001_CatalogEntry_Unstructured_en_US"
            path="/MC_10001/en_US/CatalogEntry/unstructured"
            serverName="DedicatedSearch" synonym="/conf/synonyms.txt"/>
        <_config:core catalog="10001" indexName="CatalogEntry"
            language="ro_RO" name="MC_10001_CatalogEntry_ro_RO"
            path="/MC_10001/ro_RO/CatalogEntry"
            serverName="DedicatedRomanian" synonym="/conf/synonyms.txt"/>
        <_config:core catalog="10001" indexName="UnstructuredContent"
            language="ro_RO"
            name="MC_10001_CatalogEntry_Unstructured_ro_RO"
            path="/MC_10001/ro_RO/CatalogEntry/unstructured"
            serverName="DedicatedRomanian" synonym="/conf/synonyms.txt"/>
    </_config:cores>
```

Deploy the newly modified `wc-search.xml` to the ear by using the single file update in the WebSphere Application Server console. After this configuration, query the store both in English and Romanian to ensure that the queries are being sent to the proper search server. You can check the `SystemOut.log` on each search server to ensure that the query is properly received.

7

Index design and data load

In this chapter, we provide an overview of the design and management of Solr indexes, an overview of the IBM WebSphere Commerce V7.0 catalog index, and an example of how to customize the catalog index to handle inventory data. Additionally, we demonstrate using Solr's Data Import Handler (DIH) to process delta updates using XML documents.

7.1 Solr V1.3 indexes

Solr is a stand-alone enterprise search server with a Representational State Transfer (REST)-like application programming interface (API). You put documents in the Solr server, which is called *indexing*, via XML, JavaScript Object Notation (JSON), or binary over HTTP. You query it via HTTP GET and receive XML, JSON, or binary results. Solr uses the Lucene search engine API.

7.2 IBM WebSphere Commerce V7.0 index

WebSphere Commerce V7.0 search uses one index configuration, the catalog index. Multiple catalog instances, or *cores*, exist in a production environment. For our exercise, we deploy the index inside the WebSphere Commerce Developer Edition 7.0 in embedded mode.

7.2.1 Catalogs

Search tools to work with indexes require a master catalog to be used, and they index the sales catalogs correctly. When a Solr query is received, you must always use a master catalog and not a sales catalog. You still have full use of sales catalogs for search driven navigation and searching, by continuing to use `catalogId=<sales catalog id>` when making requests. Commerce adds a filter query, `fq=catalog_id:<catalogId>`. This filter query causes the response to contain only documents in that catalog.

7.2.2 Setting up the search index

In this section, we run the `setupSearchIndex.bat` to set up the catalog index. The `setupSearchIndex.bat` creates the configuration files that are needed to preprocess the index data. Follow these steps:

1. Start the WebSphere Commerce Developer Edition test server (see Figure 7-1 on page 181). Complete these tasks:
   a. Start Rational® Application Developer for WebSphere Software.
   b. Select the **Servers** tab.
   c. Select **WebSphere Commerce Test Server**.
   d. Right-click, and then, select **Start**.

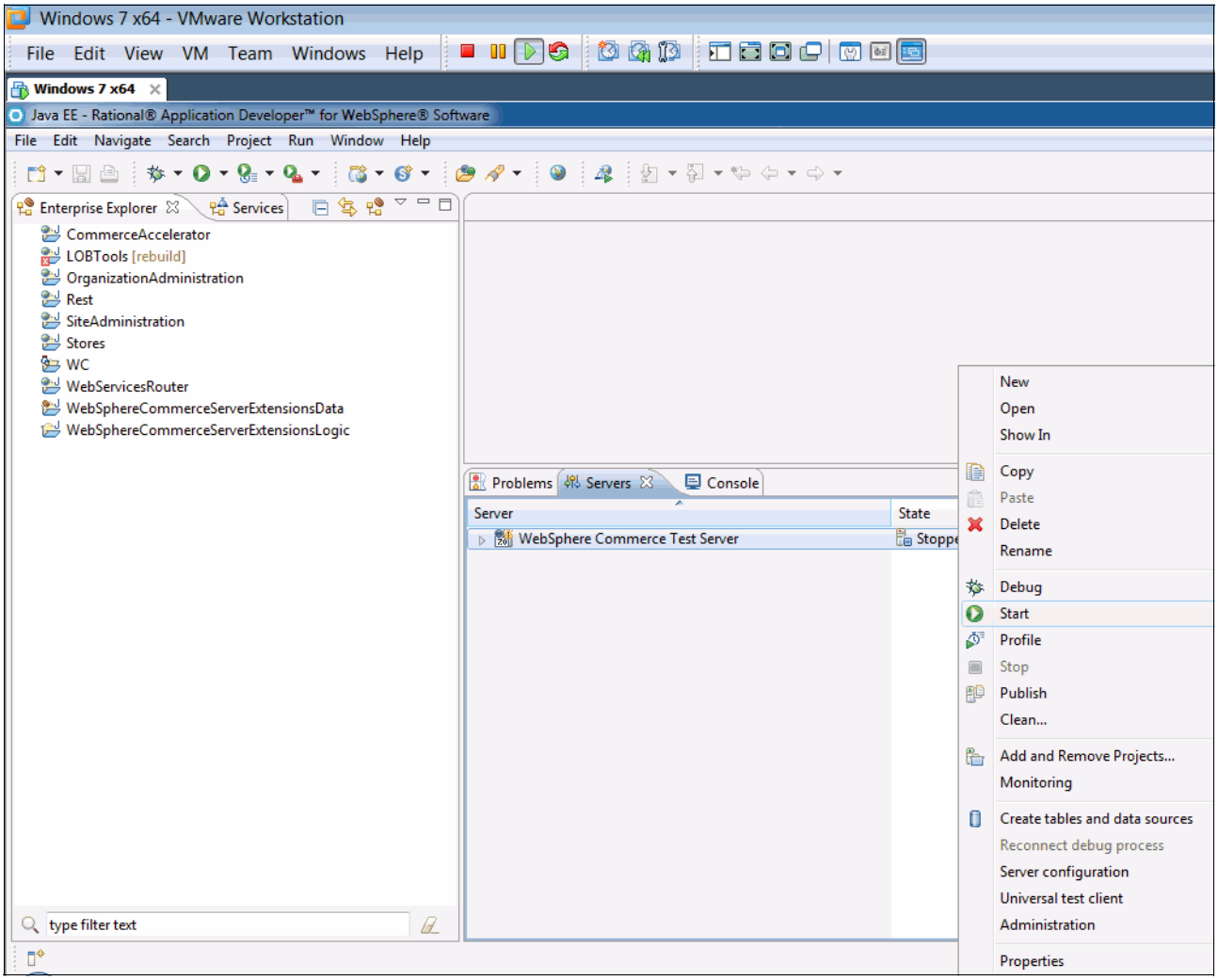


Figure 7-1 Starting WebSphere Commerce Developer Edition test server

2. Open the command window. See Figure 7-2.

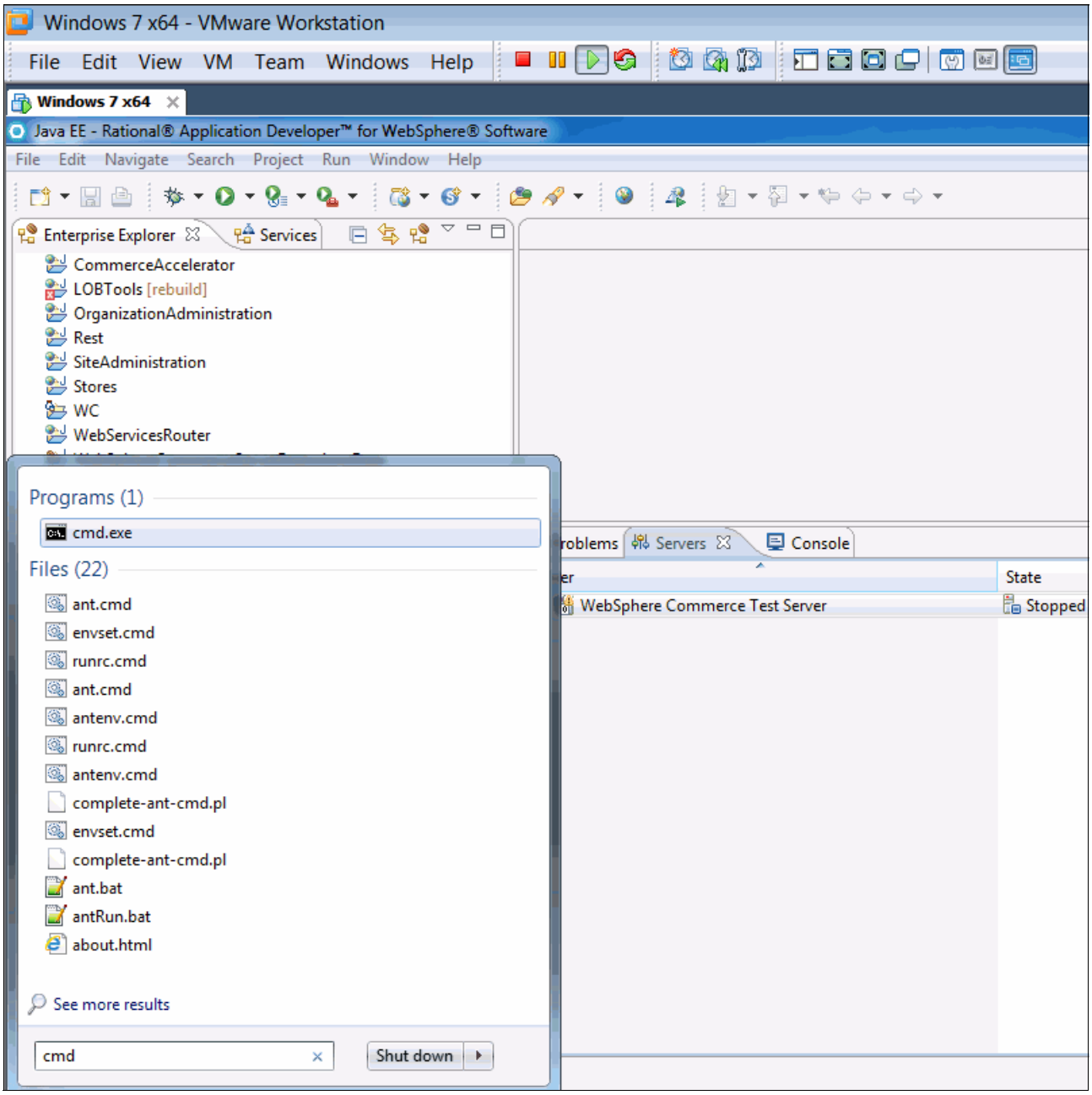


Figure 7-2 Run cmd.exe

3. Change the directory:

```
c:\cd \IBM\WCDE_ENT70\components\foundation\subcomponents\search\bin
```

4. Run `setupSearchIndex.bat`:

```
C:\IBM\WCDE_ENT70\components\foundation\subcomponents\search\bin\setupSearchInd
ex.bat -masterCatalogId 10451
```

5. Ensure that `setupSearchIndex.sh` ran successfully:
   a. Check the log file `wc-search-index-setup.log` for errors:
      i. Change the directory:

         ```
         c:\ cd \IBM\WCDE_ENT70\components\foundation\subcomponents\search\log
         ```

      ii. View the log file:

         ```
         notepad wc-search-index-setup.log
         ```

      iii. Verify that the following assets were created (see Figure 7-3 and Figure 7-4 on page 184):

         ```
         cd /opt/IBM/WebSphere/CommerceServer70/instances/demo/search/solr/home
         ```

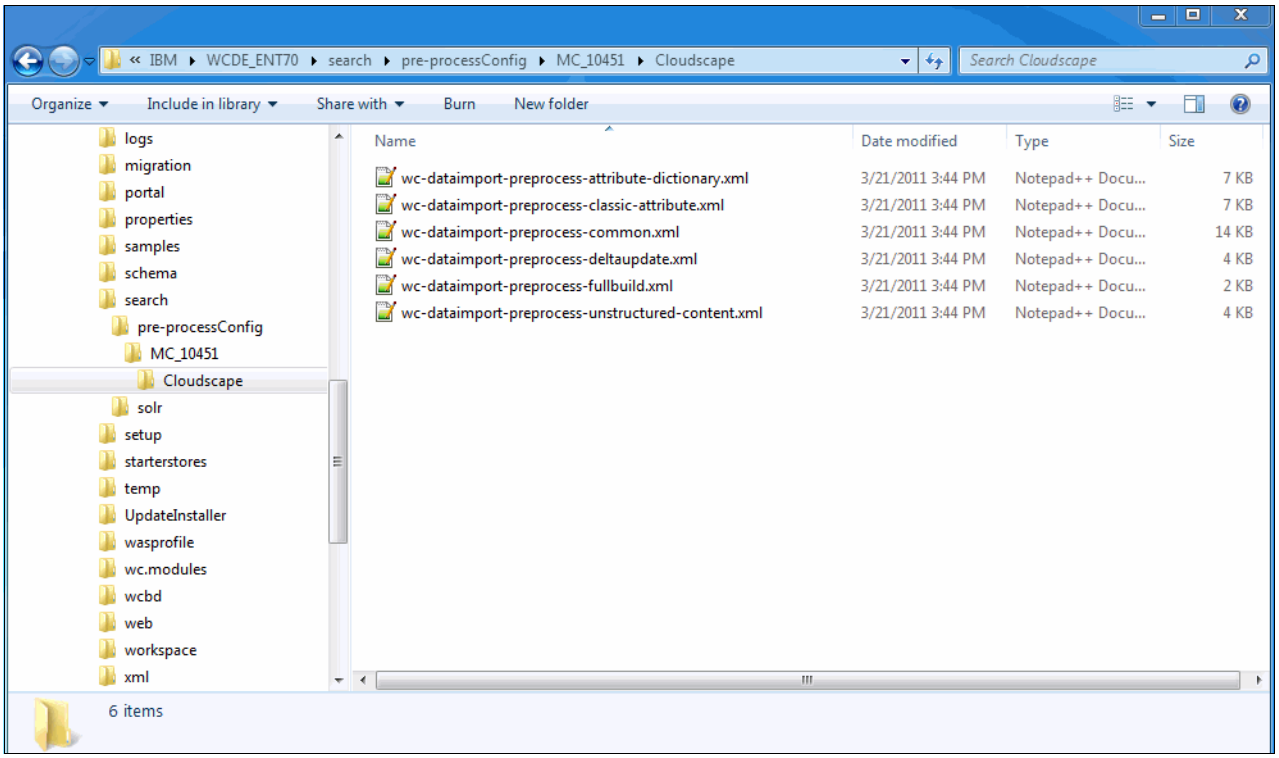


Figure 7-3 Preprocess search assets

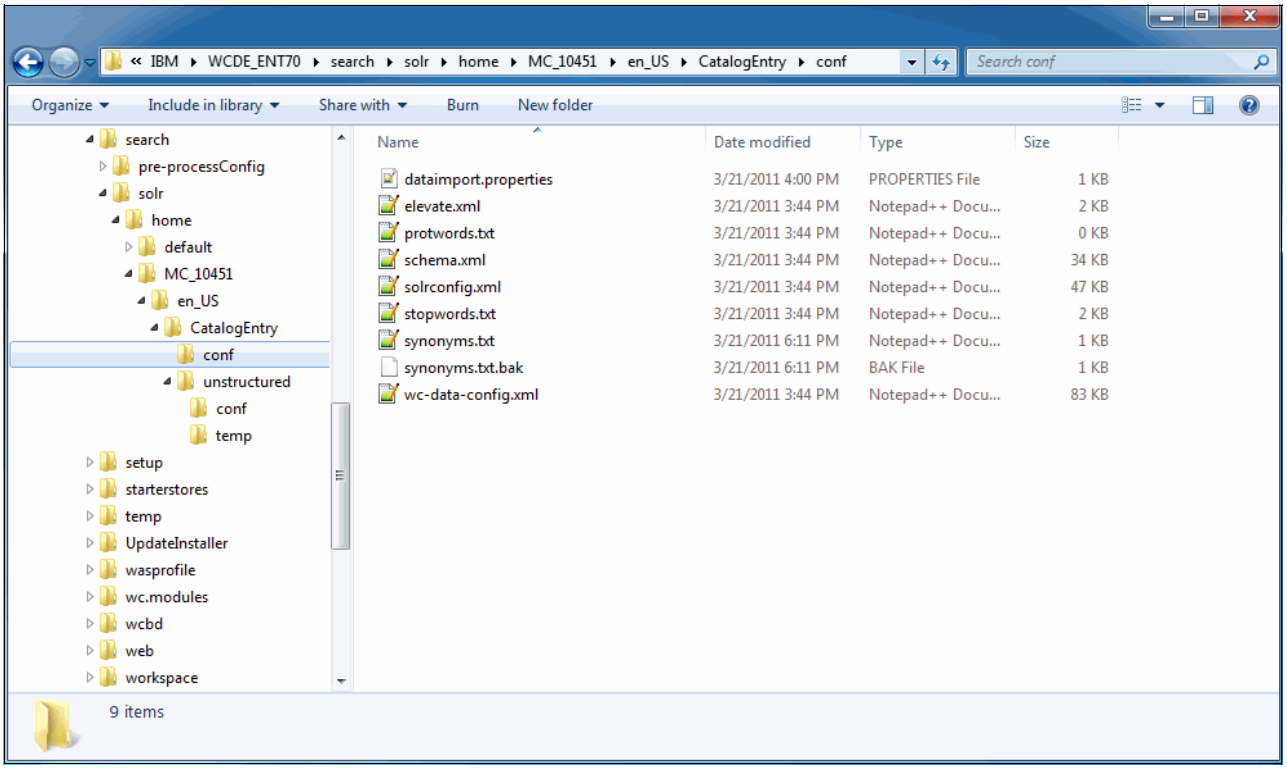


Figure 7-4 Index search assets

b. Verify the DB2 assets:

```
select * from SRCHATTR;
select * from SRCHATTRPROP;
select * from SRCHCONF;
select * from SRCHSTATS;
select * from SRCHTERM;
select * from SRCHTERMASSOCS;
```

6. Restart the WebSphere Commerce Developer Edition V7 test server.

Additional Information: For more information, go to this website:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchsetuplocal.htm

7.2.3 Preprocessing the index data

Preprocess the search index data to prepare your WebSphere Commerce V7.0 data for indexing. The preprocess utility, `di-preprocess.bat`, extracts, flattens, and imports your data into a set of temporary tables inside the WebSphere Commerce V7.0 database. This preprocessed data is then used by the index building utility, `di-buildindex.bat`, to create the indexes using Solr's Data Import Handler (DIH). The preprocessed data in the temporary tables is from the WebSphere Commerce V7.0 base schema. Follow these steps:

1. Start the WebSphere Commerce Developer Edition test server (see step 1 on page 180).
2. Change the directory:

```
c:\cd \IBM\WCDE_ENT70\bin
```

3. Run `di-preprocess.bat`:

   ```
   C:\IBM\WCDE_ENT70\bin\di-preprocess.bat
   "C:\IBM\WCDE_ENT70\search\pre-processConfig\MC_10051\Cloudscape" -fullbuild
   true -localename en_US
   ```

4. Ensure that `di-preprocess.bat` ran successfully:

   a. Check the log file `wc-search-index-setup.log` for errors:

      i. Change the directory:

      ```
      c:\cd \IBM\WCDE_ENT70\logs
      ```

      ii. View the log file:

      ```
      notepad wc-dataimport-preprocess.log
      ```

   b. Verify that the TI_<*NAME*>_0 tables were created and populated:

      ```
      http://localhost/webapp/wcs/admin/servlet/db.jsp
      ```

Additional Information: For more information, go to this website:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchbuildpre.htm

7.2.4 Building the search index

You build the WebSphere Commerce search index using the index building utility, `di-buildindex.bat`. It is a wrapping utility that updates the information in the master index using Solr's Data Import Handler (DIH) service to build the index, either partially through delta index updates or completely through full-index builds. When there are multiple indexes, for example, each language using its own separate index, the index is built multiple times. Follow these steps:

1. Start the WebSphere Commerce Developer Edition V7 test server (see step 1 on page 180).
2. Change the directory:

   ```
   c:\cd \IBM\WCDE_ENT70\bin
   ```

3. Run `di-buildindex.bat`:

   ```
   C:\IBM\WCDE_ENT70\bin\di-buildindex.bat -masterCatalogId 10051 -localename
   en_US
   ```

   Important: Running `di-buildindex.bat` with a non-master `catalogId` raises this exception:

   ```
   com.ibm.commerce.foundation.dataimport.exception.DataImportApplicationExcept
   ion: An error occurred while setting up the search configuration.
   ```

4. Ensure that `di-buildindex.bat` ran successfully:

   a. Change the directory:

   ```
   c:\cd \IBM\WCDE_ENT70\logs
   ```

   b. View the log file (see Example 7-1 on page 186):

   ```
   notepad wc-dataimport-preprocess.log
   ```

Important: Do not skip this step:

- The `di-buildindex.bat` utility overwrites the preprocess log file.
- Before running `di-buildindex.bat`, back up `wc-dataimport-preprocess.log`.

Example 7-1 wc-dataimport-preprocess.log

```
Apr 13, 2011 11:25:56 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
logStartDate(Date)
INFO: Data import process started:Wed Apr 13 11:25:56 EDT 2011
Apr 13, 2011 11:25:56 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
buildSearchConfigFromSearchConfigFileJ2SE
INFO: Build search configuration from configuration file:
c:\IBM\WCDE_E~1\workspace\WC\xml\config\com.ibm.commerce.catalog-fep\wc-search.xml.
Apr 13, 2011 11:25:56 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
executeDIH
INFO: The Solr server is localhost, the port is 80.
Apr 13, 2011 11:25:57 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
executeDIH
INFO: Solr server core MC_10051_CatalogEntry_Unstructured_en_US status is initialized.
Apr 13, 2011 11:25:57 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
executeDIH
INFO: Solr server core MC_10051_CatalogEntry_en_US status is initialized.
Apr 13, 2011 11:26:07 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
fullDataImport
INFO: Solr server core MC_10051_CatalogEntry_Unstructured_en_US status is
{responseHeader={status=0,QTime=0},initArgs={defaults={config=wc-data-config.xml}},command=statu
s,status=idle,importResponse=,statusMessages={Total Requests made to DataSource=1, Total Rows
Fetched=8, Total Documents Skipped=0, Full Dump Started=2011-04-13 11:25:57, =Indexing
completed. Added/Updated: 4 documents. Deleted 0 documents., Committed=2011-04-13 11:25:57,
Optimized=2011-04-13 11:25:57, Total Documents Processed=4, Time taken =0:0:0.313},WARNING=This
response format is experimental.  It is likely to change in the future.}.
Apr 13, 2011 11:26:07 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
fullDataImport
INFO:
Apr 13, 2011 11:26:17 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
fullDataImport
INFO: Solr server core MC_10051_CatalogEntry_en_US status is
{responseHeader={status=0,QTime=0},initArgs={defaults={config=wc-data-config.xml}},command=statu
s,status=idle,importResponse=,statusMessages={Total Requests made to DataSource=3, Total Rows
Fetched=693, Total Documents Skipped=0, Full Dump Started=2011-04-13 11:26:08, =Indexing
completed. Added/Updated: 687 documents. Deleted 0 documents., Committed=2011-04-13 11:26:09,
Optimized=2011-04-13 11:26:09, Total Documents Processed=687, Total Documents Failed=2, Time
taken =0:0:1.735},WARNING=This response format is experimental.  It is likely to change in the
future.}.
Apr 13, 2011 11:26:17 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
fullDataImport
INFO:
Apr 13, 2011 11:26:17 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
logExitCode
INFO:
------------------------------------------------
Apr 13, 2011 11:26:17 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
logExitCode(int)
```

```
INFO:
Program exiting with exit code: 0.
Data import process completed successfully with no errors.
Apr 13, 2011 11:26:17 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
logExitCode
INFO:
-----------------------------------------------
Apr 13, 2011 11:26:17 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
logEndDateAndTime
INFO: Data import process ended:Wed Apr 13 11:26:17 EDT 2011
Apr 13, 2011 11:26:17 AM com.ibm.commerce.foundation.dataimport.process.DataImportProcessorMain
logEndDateAndTime
INFO: Data import process completed in 20.593 seconds.
```

c. Verify that the `MC_10051_CatalogEntry_en_US` index is on the disk. See Figure 7-5.

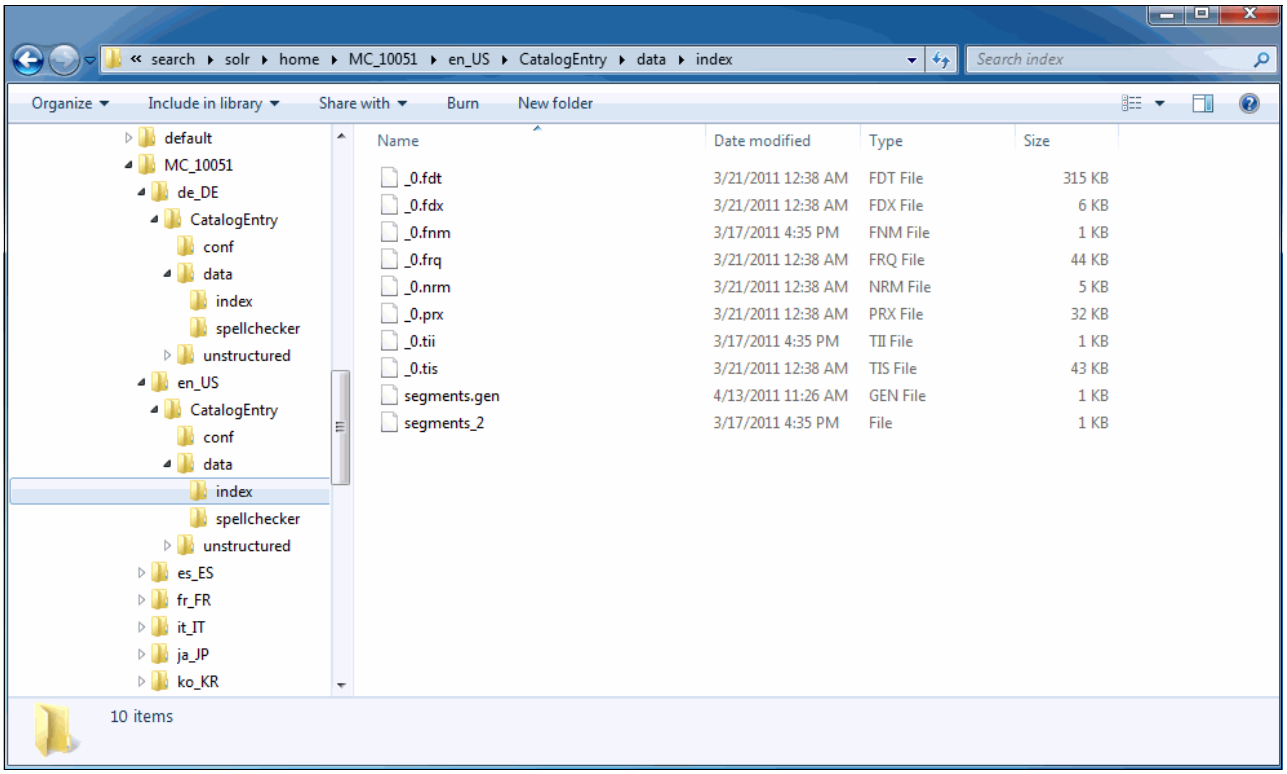


Figure 7-5 MC_10051_CatalogEntry_en_US structured index

d. Execute the query and view the result (Example 7-2):

`http://localhost/solr/MC_10051_CatalogEntry_en_US/select?q=red&fl=*`

Example 7-2 Query response

```
<?xml version="1.0" encoding="utf-8"?>
<response>
   <lst name="responseHeader">
   <int name="status">0</int>
   <int name="QTime">15</int>
   <lst name="params">
     <str name="fl">*</str>
     <str name="q">red</str>
```

```
   </lst></lst>
   <result name="response" start="0" numFound="24">
      <doc>
         <int name="buyable">1</int>
         <arr name="catalog_id">
           <long>10051</long>
         </arr>
         <long name="catentry_id">10343</long>
         <str name="catenttype_id_ntk_cs">ProductBean</str>
         <str name="fullImage">
            images/catalog/kitchenware/kitchenware_160x160/cm_14.jpg
         </str>
         <long name="member_id">7000000000000000101</long>
         <str name="mfName">Enzi</str>
         <str name="mfName_ntk">Enzi</str>
         <str name="mfName_ntk_cs">Enzi</str>
         <str name="name">Enzi Espresso Machine, Red</str>
         <arr name="parentCatgroup_id_facet">
           <str>10051_10074</str>
         </arr>
         <arr name="parentCatgroup_id_search">
           <str>10051_10074</str>
           <str>10051_10053</str>
         </arr>
         <str name="partNumber_ntk">EI-01R</str>
         <float name="price_USD">179.99</float>
         <int name="published">1</int>
         <str name="shortDescription">A classy looking, red stainless steel Pump Espresso
machine.</str>
         <int name="storeent_id">10051</int>
         <str name="thumbnail">images/catalog/kitchenware/kitchenware_70x70/cm_14.jpg</str>
      </doc>
      <!-- remaining documents removed -->
   </result>
</response>
```

Additional Information: For more information, go to this website:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchbuildindex.htm

7.2.5 Maintenance

The WebSphere Commerce V7.0 search index falls out of synchronization with the latest production data over time. To maintain the accuracy of the search result data, you must perform re-indexing during normal business operation.

Additional Information: For more information, go to this website:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.developer.doc/concepts/csdsearchcontentstructureindex.htm

7.3 Scenario: Display only products with inventory

A common business requirement is to display only products that have inventory on the storefront. Using WebSphere Commerce V7.0 search requires performing an index update. There is always a period of time, *t1*, when products in the catalog index potentially have no inventory. Our implementation minimizes *t1* and gives the shopper a better shopping experience. We describe the process:

1. An order is placed for product X.
2. The order is fulfilled for product X. Product X now has 0 inventory.
3. Start *t1*.
4. Perform delta re-indexing:
   a. For each product that has 0 inventory, generate and fire a ChangeCatalogEntry event.
   b. Generate and fire an index synchronization event.
5. Stop *t1*.

7.3.1 Adding the inventory field to the catalog index

For this solution, we customize the WebSphere Commerce V7.0 index by adding an additional field for inventory. We must flatten out the catalog entry/inventory data, which involves creating a custom preprocessor and custom temporary index tables.

Customizing: Your table name must start with an `X` to avoid conflict with the default WebSphere Commerce tables, for example, XI_INVENTORY_0.

Follow these steps:

1. Configure the search preprocessor:
   a. Create a new custom preprocess configuration file:
      i. Open the Notepad.
      ii. Select **File** → **Save As**:

      ```
      c:\IBM\WCDE_ENT70\search\pre-processConfig\MC_10051\Cloudscape\wc-dataimport-preprocess-custom-inventory.xml
      ```

      iii. Add the code (Example 7-3 on page 190).

Example 7-3 wc-dataimport-preprocess-custom-inventory.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<_config:DIHPreProcessConfig
    xmlns:_config="http://www.ibm.com/xmlns/prod/commerce/foundation/config"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ibm.com/xmlns/prod/commerce/foundation/config ../../xsd/wc-dataimport-preprocess.xsd">


<!-- one level PRODUCT_ITEM catentry parent -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
    <_config:table definition="CREATE TABLE XI_DPCATENTRY_PRODUCT_ITEM_0 (CATENTRY_ID BIGINT NOT NULL, CATENTRY_ID_PARENT BIGINT, PRIMARY
KEY (CATENTRY_ID))" name="XI_DPCATENTRY_PRODUCT_ITEM_0"/>
    <_config:query sql="SELECT CATENTRY_ID_CHILD, CATENTRY_ID_PARENT FROM CATENTREL R, TI_CATENTRY_0 C WHERE R.CATENTRY_ID_CHILD =
C.CATENTRY_ID AND R.CATRELTYPE_ID='PRODUCT_ITEM' ORDER BY CATENTRY_ID_CHILD"/>
    <_config:mapping>
        <_config:key queryColumn="CATENTRY_ID_CHILD" tableColumn="CATENTRY_ID"/>
        <_config:column-mapping>
            <_config:column-column-mapping>
                <_config:column-column queryColumn="CATENTRY_ID_PARENT" tableColumn="CATENTRY_ID_PARENT" />
            </_config:column-column-mapping>
        </_config:column-mapping>
    </_config:mapping>
</_config:data-processing-config>

<!-- one level BUNDLE_COMPONENT catentry parent -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
    <_config:table definition="CREATE TABLE XI_DPCATENTRY_BUNDLE_COMPONENT_0 (CATENTRY_ID BIGINT NOT NULL, CATENTRY_ID_PARENT BIGINT,
PRIMARY KEY (CATENTRY_ID))" name="XI_DPCATENTRY_BUNDLE_COMPONENT_0"/>
    <_config:query sql="SELECT CATENTRY_ID_CHILD, CATENTRY_ID_PARENT FROM CATENTREL R, TI_CATENTRY_0 C WHERE R.CATENTRY_ID_CHILD =
C.CATENTRY_ID AND R.CATRELTYPE_ID='BUNDLE_COMPONENT' ORDER BY CATENTRY_ID_CHILD"/>
    <_config:mapping>
        <_config:key queryColumn="CATENTRY_ID_CHILD" tableColumn="CATENTRY_ID"/>
        <_config:column-mapping>
            <_config:column-column-mapping>
                <_config:column-column queryColumn="CATENTRY_ID_PARENT" tableColumn="CATENTRY_ID_PARENT" />
            </_config:column-column-mapping>
        </_config:column-mapping>
    </_config:mapping>
</_config:data-processing-config>

<!-- ItemBean inventory -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
  <!-- leaving ffmcenterId out as any inventory is inventory -->
  <_config:table definition="CREATE TABLE XI_ITEM_INVENTORY_0 (CATENTRY_ID BIGINT NOT NULL, STORE_ID BIGINT NOT NULL, QUANTITY BIGINT,
PRIMARY KEY (CATENTRY_ID,STORE_ID))" name="XI_ITEM_INVENTORY_0"/>
  <_config:query sql="SELECT I.CATENTRY_ID, I.STORE_ID, SUM( CAST(I.QUANTITY as BIGINT) ) quantity FROM INVENTORY I,
XI_DPCATENTRY_PRODUCT_ITEM_0 CE WHERE I.STORE_ID IN ( SELECT STOREENT_ID from STORECAT WHERE CATALOG_ID=10001 ) AND I.CATENTRY_ID =
CE.CATENTRY_ID GROUP BY I.CATENTRY_ID, I.STORE_ID ORDER BY I.CATENTRY_ID"/>
  <_config:mapping>
    <_config:key queryColumn="CATENTRY_ID" tableColumn="CATENTRY_ID"/>
    <_config:column-mapping>
      <_config:column-column-mapping>
    <_config:column-column queryColumn="STORE_ID" tableColumn="STORE_ID" />
    <_config:column-column queryColumn="QUANTITY" tableColumn="QUANTITY" />
      </_config:column-column-mapping>
    </_config:column-mapping>
  </_config:mapping>
</_config:data-processing-config>

<!-- ProductBean inventory: this is the inventory of the ItemBean(s) with a PRODUCT_ITEM  relationship -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
  <!-- leaving ffmcenterId out as any inventory is inventory -->
  <_config:table definition="CREATE TABLE XI_PRODUCT_INVENTORY_0 (CATENTRY_ID BIGINT NOT NULL, CATENTRY_ID_PARENT VARCHAR(512), QUANTITY
BIGINT, PRIMARY KEY (CATENTRY_ID))" name="XI_PRODUCT_INVENTORY_0"/>
  <_config:query sql="SELECT T.CATENTRY_ID, T.CATENTRY_ID_PARENT, X.QUANTITY FROM XI_ITEM_INVENTORY_0 X INNER JOIN
XI_DPCATENTRY_PRODUCT_ITEM_0 T ON X.CATENTRY_ID = T.CATENTRY_ID"/>
  <_config:mapping>
    <_config:key queryColumn="CATENTRY_ID" tableColumn="CATENTRY_ID"/>
    <_config:column-mapping>
      <_config:column-column-mapping>
    <_config:column-column queryColumn="CATENTRY_ID_PARENT" tableColumn="CATENTRY_ID_PARENT" />
    <_config:column-column queryColumn="QUANTITY" tableColumn="QUANTITY" />
      </_config:column-column-mapping>
    </_config:column-mapping>
  </_config:mapping>
</_config:data-processing-config>

<!-- BundleBean inventory: this is the inventory of the ItemBean(s) with a BUNDLE_COMPONENT  relationship -->
```

```
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
  <!-- leaving ffmcenterId out as any inventory is inventory -->
  <_config:table definition="CREATE TABLE XI_BUNDLE_INVENTORY_0 (CATENTRY_ID BIGINT NOT NULL, CATENTRY_ID_PARENT VARCHAR(512), QUANTITY
BIGINT, PRIMARY KEY (CATENTRY_ID))" name="XI_BUNDLE_INVENTORY_0"/>
  <_config:query sql="SELECT T.CATENTRY_ID, T.CATENTRY_ID_PARENT, X.QUANTITY FROM XI_ITEM_INVENTORY_0 X INNER JOIN
XI_DPCATENTRY_BUNDLE_COMPONENT_0 T ON X.CATENTRY_ID = T.CATENTRY_ID"/>
  <_config:mapping>
    <_config:key queryColumn="CATENTRY_ID" tableColumn="CATENTRY_ID"/>
    <_config:column-mapping>
      <_config:column-column-mapping>
    <_config:column-column queryColumn="CATENTRY_ID_PARENT" tableColumn="CATENTRY_ID_PARENT" />
    <_config:column-column queryColumn="QUANTITY" tableColumn="QUANTITY" />
      </_config:column-column-mapping>
    </_config:column-mapping>
  </_config:mapping>
</_config:data-processing-config>
</_config:DIHPreProcessConfig>
```

b. Preprocess the search index data. See 7.2.3, “Preprocessing the index data” on page 184.

c. Verify the custom index tables with SQL:

- `select * from XI_DPCATENTRY_PRODUCT_ITEM_0;`
- `select * from XI_DPCATENTRY_BUNDLE_COMPONENT_0;`
- `select * from XI_ITEM_INVENTORY_0;`
- `select * from XI_PRODUCT_INVENTORY_0;`
- `select * from XI_BUNDLE_INVENTORY_0;`

Troubleshooting: The following exception sometimes occurs when running the `di-preprocess.bat` utility while any of the TI_*XXXX* or XI_*XXXX* tables are in use by another application, for example, a query in the DB2 V9.7 Control Center:

```
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain
handleExecutionException

SEVERE: CWFDIH0002: An SQL exception was caught. The following error occurred:
DB2 SQL Error: SQLCODE=-601, SQLSTATE=42710,
SQLERRMC=DB2INST1.<TABLE_NAME>;TABLE, DRIVER=4.7.85.
```

2. Configure the Data Import Handler mapping:

a. Open the Notepad.

b. Click **File** → **Open** and open this file:

```
c:\IBM\WCDE_ENT70\search\solr\home\MC_10051\en_US\CatalogEntry\conf\schema.x
ml
```

c. Find the `//schema[@name='WebSphereCommerceSolrSchema']/fields` node.

d. Insert the new field node (see Example 7-4).

Example 7-4 Inventory field definition

```
<!-- Inventory's quantity: map to table: INVENTORY -->
  <field name="xquantity" type="long" indexed="true" stored="true" multiValued="false" />
```

e. Save your changes.

f. Select **File** → **Open** and open this file:

```
C:\IBM\WCDE_ENT70\search\solr\home\MC_10051\en_US\CatalogEntry\conf\wc-data-
config.xml
```

g. Find the `//dataConfig/document/entity[@name='Product']` node:

i. Update the query attribute value (see Example 7-5).

Example 7-5 Query attribute value

```
SELECT CATENTRY.CATENTRY_ID,CATENTRY.MEMBER_ID,CATENTRY.CATENTTYPE_ID,CATENTRY.PARTNUMBER,
CATENTRY.MFPARTNUMBER,CATENTRY.MFNAME, CATENTRY.BUYABLE, STORECENT.STOREENT_ID,
CATENTDESC.NAME,CATENTDESC.SHORTDESCRIPTION,CATENTDESC.THUMBNAIL,CATENTDESC.FULLIMAGE,
CATENTDESC.KEYWORD, CATENTDESC.PUBLISHED, TI_DPGROUP.CATGROUP DPCATGROUP,
TI_APGROUP.CATGROUPS APCATGROUP, TI_PRODUCTSET.PRODUCTSET,
TI_OFFERPRICE.PRICE_USD, TI_OFFERPRICE.PRICE_EUR, TI_OFFERPRICE.PRICE_CAD,
TI_OFFERPRICE.PRICE_CNY,
TI_OFFERPRICE.PRICE_JPY, TI_OFFERPRICE.PRICE_KRW, TI_OFFERPRICE.PRICE_BRL,
TI_OFFERPRICE.PRICE_TWD,
TI_OFFERPRICE.PRICE_PLN, TI_OFFERPRICE.PRICE_PLN, TI_OFFERPRICE.PRICE_RON,
TI_OFFERPRICE.PRICE_EGP,
TI_OFFERPRICE.PRICE_GBP,TI_DPCATENTRY.CATENTRY_PARENT,
TI_CATALOG.CATALOG PARENT_CATALOG_ID,

XI_PRODUCT_INVENTORY.QUANTITY XQUANTITY,

TI_CASTB1.CAS_F1 CAS_F1ATTR, TI_CASTB1.CAS_F2 CAS_F2ATTR, TI_CASTB1.CAS_F3
CAS_F3ATTR,TI_CASTB1.CAS_F4 CAS_F4ATTR,TI_CASTB1.CAS_F5 CAS_F5ATTR,
TI_CASTB1.CAS_F6 CAS_F6ATTR,TI_CASTB1.CAS_F7 CAS_F7ATTR,TI_CASTB1.CAS_F8
CAS_F8ATTR,TI_CASTB1.CAS_F9 CAS_F9ATTR,TI_CASTB1.CAS_F10 CAS_F10ATTR,
TI_CASTB1.CAS_F11 CAS_F11ATTR,TI_CASTB1.CAS_F12 CAS_F12ATTR,TI_CASTB1.CAS_F13
CAS_F13ATTR,TI_CASTB1.CAS_F14 CAS_F14ATTR,TI_CASTB1.CAS_F15 CAS_F15ATTR,
TI_CASTB1.CAS_F16 CAS_F16ATTR,TI_CASTB1.CAS_F17 CAS_F17ATTR, TI_CASTB1.CAS_F18
CAS_F18ATTR,TI_CASTB1.CAS_F19 CAS_F19ATTR,TI_CASTB1.CAS_F20 CAS_F20ATTR,
TI_CASTB1.CAS_F21 CAS_F21ATTR,TI_CASTB1.CAS_F22 CAS_F22ATTR,TI_CASTB1.CAS_F23
CAS_F23ATTR,TI_CASTB1.CAS_F24 CAS_F24ATTR,TI_CASTB1.CAS_F25  CAS_F21ATTR,
TI_CASTB1.CAS_F26  CAS_F26ATTR,TI_CASTB1.CAS_F27  CAS_F27ATTR,TI_CASTB1.CAS_F28
CAS_F28ATTR,TI_CASTB1.CAS_F29  CAS_F29ATTR,TI_CASTB1.CAS_F30  CAS_F30ATTR,
TI_CAITB1.CAI_F1,TI_CAITB1.CAI_F2,TI_CAITB1.CAI_F3,TI_CAITB1.CAI_F4,TI_CAITB1.CAI_F5,
TI_CAITB1.CAI_F6,TI_CAITB1.CAI_F7,TI_CAITB1.CAI_F8,TI_CAITB1.CAI_F9,TI_CAITB1.CAI_F10,
TI_CAFTB1.CAF_F1, TI_CAFTB1.CAF_F2, TI_CAFTB1.CAF_F3, TI_CAFTB1.CAF_F4, TI_CAFTB1.CAF_F5,
TI_CAFTB1.CAF_F6, TI_CAFTB1.CAF_F7, TI_CAFTB1.CAF_F8, TI_CAFTB1.CAF_F9, TI_CAFTB1.CAF_F10,

TI_ADSTB1.ADS_F1 ADS_F1ATTR, TI_ADSTB1.ADS_F2 ADS_F2ATTR, TI_ADSTB1.ADS_F3
ADS_F3ATTR,TI_ADSTB1.ADS_F4 ADS_F4ATTR,TI_ADSTB1.ADS_F5 ADS_F5ATTR,
TI_ADSTB1.ADS_F6 ADS_F6ATTR,TI_ADSTB1.ADS_F7 ADS_F7ATTR,TI_ADSTB1.ADS_F8
ADS_F8ATTR,TI_ADSTB1.ADS_F9 ADS_F9ATTR,TI_ADSTB1.ADS_F10 ADS_F10ATTR,
TI_ADSTB1.ADS_F11 ADS_F11ATTR,TI_ADSTB1.ADS_F12 ADS_F12ATTR,TI_ADSTB1.ADS_F13
ADS_F13ATTR,TI_ADSTB1.ADS_F14 ADS_F14ATTR,TI_ADSTB1.ADS_F15 ADS_F15ATTR,
TI_ADSTB1.ADS_F16 ADS_F16ATTR,TI_ADSTB1.ADS_F17 ADS_F17ATTR, TI_ADSTB1.ADS_F18
ADS_F18ATTR,TI_ADSTB1.ADS_F19 ADS_F19ATTR,TI_ADSTB1.ADS_F20 ADS_F20ATTR,
TI_ADSTB1.ADS_F21 ADS_F21ATTR,TI_ADSTB1.ADS_F22 ADS_F22ATTR,TI_ADSTB1.ADS_F23
ADS_F23ATTR,TI_ADSTB1.ADS_F24 ADS_F24ATTR,TI_ADSTB1.ADS_F25  ADS_F21ATTR,
TI_ADSTB1.ADS_F26  ADS_F26ATTR,TI_ADSTB1.ADS_F27  ADS_F27ATTR,TI_ADSTB1.ADS_F28
ADS_F28ATTR,TI_ADSTB1.ADS_F29  ADS_F29ATTR,TI_ADSTB1.ADS_F30  ADS_F30ATTR,
TI_ADITB1.ADI_F1,TI_ADITB1.ADI_F2,TI_ADITB1.ADI_F3,TI_ADITB1.ADI_F4,TI_ADITB1.ADI_F5,
TI_ADITB1.ADI_F6,TI_ADITB1.ADI_F7,TI_ADITB1.ADI_F8,TI_ADITB1.ADI_F9,TI_ADITB1.ADI_F10,
TI_ADFTB1.ADF_F1, TI_ADFTB1.ADF_F2, TI_ADFTB1.ADF_F3, TI_ADFTB1.ADF_F4, TI_ADFTB1.ADF_F5,
TI_ADFTB1.ADF_F6, TI_ADFTB1.ADF_F7, TI_ADFTB1.ADF_F8, TI_ADFTB1.ADF_F9, TI_ADFTB1.ADF_F10

FROM CATENTRY
```

```
INNER JOIN TI_CATENTRY_0 TI_CATENTRY ON (CATENTRY.CATENTRY_ID=TI_CATENTRY.CATENTRY_ID)
LEFT OUTER JOIN STORECENT ON (CATENTRY.CATENTRY_ID=STORECENT.CATENTRY_ID)
LEFT OUTER JOIN CATENTDESC ON (CATENTDESC.CATENTRY_ID=CATENTRY.CATENTRY_ID AND
CATENTDESC.LANGUAGE_ID=-1)
LEFT OUTER JOIN TI_DPGROUP_0 TI_DPGROUP ON (CATENTRY.CATENTRY_ID=TI_DPGROUP.CATENTRY_ID)
LEFT OUTER JOIN TI_APGROUP_0 TI_APGROUP ON (CATENTRY.CATENTRY_ID=TI_APGROUP.CATENTRY_ID)
LEFT OUTER JOIN TI_PRODUCTSET_0 TI_PRODUCTSET ON
(CATENTRY.CATENTRY_ID=TI_PRODUCTSET.CATENTRY_ID)
LEFT OUTER JOIN TI_OFFERPRICE_0 TI_OFFERPRICE ON
(CATENTRY.CATENTRY_ID=TI_OFFERPRICE.CATENTRY_ID)
LEFT OUTER JOIN TI_DPCATENTRY_0 TI_DPCATENTRY ON
(CATENTRY.CATENTRY_ID=TI_DPCATENTRY.CATENTRY_ID)
LEFT OUTER JOIN TI_CATALOG_0 TI_CATALOG ON (CATENTRY.CATENTRY_ID=TI_CATALOG.CATENTRY_ID)

LEFT OUTER JOIN XI_PRODUCT_INVENTORY_0 XI_PRODUCT_INVENTORY ON (CATENTRY.CATENTRY_ID=
CAST(XI_PRODUCT_INVENTORY.CATENTRY_ID_PARENT AS BIGINT) )

LEFT OUTER JOIN TI_CASTB1_0_1 TI_CASTB1 ON (CATENTRY.CATENTRY_ID=TI_CASTB1.CATENTRY_ID)
LEFT OUTER JOIN TI_CAITB1_0_1 TI_CAITB1 ON (CATENTRY.CATENTRY_ID=TI_CAITB1.CATENTRY_ID)
LEFT OUTER JOIN TI_CAFTB1_0_1 TI_CAFTB1 ON (CATENTRY.CATENTRY_ID=TI_CAFTB1.CATENTRY_ID)
LEFT OUTER JOIN TI_ADSTB1_0_1 TI_ADSTB1 ON (CATENTRY.CATENTRY_ID=TI_ADSTB1.CATENTRY_ID)
LEFT OUTER JOIN TI_ADITB1_0_1 TI_ADITB1 ON (CATENTRY.CATENTRY_ID=TI_ADITB1.CATENTRY_ID)
LEFT OUTER JOIN TI_ADFTB1_0_1 TI_ADFTB1 ON (CATENTRY.CATENTRY_ID=TI_ADFTB1.CATENTRY_ID)
WHERE  CATENTRY.CATENTTYPE_ID='ProductBean'
```

ii. Update the `deltaImportQuery` attribute value (see Example 7-6).

Example 7-6 deltaImportQuery attribute value

```
SELECT CATENTRY.CATENTRY_ID,CATENTRY.MEMBER_ID,CATENTRY.CATENTTYPE_ID,CATENTRY.PARTNUMBER,
CATENTRY.MFPARTNUMBER,CATENTRY.MFNAME, CATENTRY.BUYABLE,STORECENT.STOREENT_ID,
CATENTDESC.NAME,CATENTDESC.SHORTDESCRIPTION,CATENTDESC.THUMBNAIL,CATENTDESC.FULLIMAGE,
CATENTDESC.KEYWORD, CATENTDESC.PUBLISHED,TI_DPGROUP.CATGROUP DPCATGROUP,
TI_APGROUP.CATGROUPS APCATGROUP,TI_PRODUCTSET.PRODUCTSET,
TI_OFFERPRICE.PRICE_USD, TI_OFFERPRICE.PRICE_EUR, TI_OFFERPRICE.PRICE_CAD,
TI_OFFERPRICE.PRICE_CNY,
TI_OFFERPRICE.PRICE_JPY, TI_OFFERPRICE.PRICE_KRW, TI_OFFERPRICE.PRICE_BRL,
TI_OFFERPRICE.PRICE_TWD,
TI_OFFERPRICE.PRICE_PLN, TI_OFFERPRICE.PRICE_PLN, TI_OFFERPRICE.PRICE_RON,
TI_OFFERPRICE.PRICE_EGP,
TI_OFFERPRICE.PRICE_GBP,TI_DPCATENTRY.CATENTRY_PARENT,
TI_CATALOG.CATALOG PARENT_CATALOG_ID,

XI_PRODUCT_INVENTORY.QUANTITY XQUANTITY,

TI_CASTB1.CAS_F1 CAS_F1ATTR, TI_CASTB1.CAS_F2 CAS_F2ATTR, TI_CASTB1.CAS_F3
CAS_F3ATTR,TI_CASTB1.CAS_F4 CAS_F4ATTR,TI_CASTB1.CAS_F5 CAS_F5ATTR,
TI_CASTB1.CAS_F6 CAS_F6ATTR,TI_CASTB1.CAS_F7 CAS_F7ATTR,TI_CASTB1.CAS_F8
CAS_F8ATTR,TI_CASTB1.CAS_F9 CAS_F9ATTR,TI_CASTB1.CAS_F10 CAS_F10ATTR,
TI_CASTB1.CAS_F11 CAS_F11ATTR,TI_CASTB1.CAS_F12 CAS_F12ATTR,TI_CASTB1.CAS_F13
CAS_F13ATTR,TI_CASTB1.CAS_F14 CAS_F14ATTR,TI_CASTB1.CAS_F15 CAS_F15ATTR,
TI_CASTB1.CAS_F16 CAS_F16ATTR,TI_CASTB1.CAS_F17 CAS_F17ATTR, TI_CASTB1.CAS_F18
CAS_F18ATTR,TI_CASTB1.CAS_F19 CAS_F19ATTR,TI_CASTB1.CAS_F20 CAS_F20ATTR,
TI_CASTB1.CAS_F21 CAS_F21ATTR,TI_CASTB1.CAS_F22 CAS_F22ATTR,TI_CASTB1.CAS_F23
CAS_F23ATTR,TI_CASTB1.CAS_F24 CAS_F24ATTR,TI_CASTB1.CAS_F25  CAS_F21ATTR,
```

```
TI_CASTB1.CAS_F26  CAS_F26ATTR,TI_CASTB1.CAS_F27  CAS_F27ATTR,TI_CASTB1.CAS_F28
CAS_F28ATTR,TI_CASTB1.CAS_F29  CAS_F29ATTR,TI_CASTB1.CAS_F30  CAS_F30ATTR,
TI_CAITB1.CAI_F1,TI_CAITB1.CAI_F2,TI_CAITB1.CAI_F3,TI_CAITB1.CAI_F4,TI_CAITB1.CAI_F5,
TI_CAITB1.CAI_F6,TI_CAITB1.CAI_F7,TI_CAITB1.CAI_F8,TI_CAITB1.CAI_F9,TI_CAITB1.CAI_F10,
TI_CAFTB1.CAF_F1, TI_CAFTB1.CAF_F2, TI_CAFTB1.CAF_F3, TI_CAFTB1.CAF_F4, TI_CAFTB1.CAF_F5,
TI_CAFTB1.CAF_F6, TI_CAFTB1.CAF_F7, TI_CAFTB1.CAF_F8, TI_CAFTB1.CAF_F9, TI_CAFTB1.CAF_F10,
TI_ADSTB1.ADS_F1 ADS_F1ATTR, TI_ADSTB1.ADS_F2 ADS_F2ATTR, TI_ADSTB1.ADS_F3
ADS_F3ATTR,TI_ADSTB1.ADS_F4 ADS_F4ATTR,TI_ADSTB1.ADS_F5 ADS_F5ATTR,
TI_ADSTB1.ADS_F6 ADS_F6ATTR,TI_ADSTB1.ADS_F7 ADS_F7ATTR,TI_ADSTB1.ADS_F8
ADS_F8ATTR,TI_ADSTB1.ADS_F9 ADS_F9ATTR,TI_ADSTB1.ADS_F10 ADS_F10ATTR,
TI_ADSTB1.ADS_F11 ADS_F11ATTR,TI_ADSTB1.ADS_F12 ADS_F12ATTR,TI_ADSTB1.ADS_F13
ADS_F13ATTR,TI_ADSTB1.ADS_F14 ADS_F14ATTR,TI_ADSTB1.ADS_F15 ADS_F15ATTR,
TI_ADSTB1.ADS_F16 ADS_F16ATTR,TI_ADSTB1.ADS_F17 ADS_F17ATTR, TI_ADSTB1.ADS_F18
ADS_F18ATTR,TI_ADSTB1.ADS_F19 ADS_F19ATTR,TI_ADSTB1.ADS_F20 ADS_F20ATTR,
TI_ADSTB1.ADS_F21 ADS_F21ATTR,TI_ADSTB1.ADS_F22 ADS_F22ATTR,TI_ADSTB1.ADS_F23
ADS_F23ATTR,TI_ADSTB1.ADS_F24 ADS_F24ATTR,TI_ADSTB1.ADS_F25  ADS_F21ATTR,
TI_ADSTB1.ADS_F26  ADS_F26ATTR,TI_ADSTB1.ADS_F27  ADS_F27ATTR,TI_ADSTB1.ADS_F28
ADS_F28ATTR,TI_ADSTB1.ADS_F29  ADS_F29ATTR,TI_ADSTB1.ADS_F30  ADS_F30ATTR,
TI_ADITB1.ADI_F1,TI_ADITB1.ADI_F2,TI_ADITB1.ADI_F3,TI_ADITB1.ADI_F4,TI_ADITB1.ADI_F5,
TI_ADITB1.ADI_F6,TI_ADITB1.ADI_F7,TI_ADITB1.ADI_F8,TI_ADITB1.ADI_F9,TI_ADITB1.ADI_F10,
TI_ADFTB1.ADF_F1, TI_ADFTB1.ADF_F2, TI_ADFTB1.ADF_F3, TI_ADFTB1.ADF_F4, TI_ADFTB1.ADF_F5,
TI_ADFTB1.ADF_F6, TI_ADFTB1.ADF_F7, TI_ADFTB1.ADF_F8, TI_ADFTB1.ADF_F9, TI_ADFTB1.ADF_F10

FROM CATENTRY
INNER JOIN TI_CATENTRY_O TI_CATENTRY ON (CATENTRY.CATENTRY_ID=TI_CATENTRY.CATENTRY_ID)
LEFT OUTER JOIN STORECENT ON (CATENTRY.CATENTRY_ID=STORECENT.CATENTRY_ID)
LEFT OUTER JOIN CATENTDESC ON (CATENTDESC.CATENTRY_ID=CATENTRY.CATENTRY_ID AND
CATENTDESC.LANGUAGE_ID=-1)
LEFT OUTER JOIN TI_DPGROUP_O TI_DPGROUP ON (CATENTRY.CATENTRY_ID=TI_DPGROUP.CATENTRY_ID)
LEFT OUTER JOIN TI_APGROUP_O TI_APGROUP ON (CATENTRY.CATENTRY_ID=TI_APGROUP.CATENTRY_ID)
LEFT OUTER JOIN TI_PRODUCTSET_O TI_PRODUCTSET ON
(CATENTRY.CATENTRY_ID=TI_PRODUCTSET.CATENTRY_ID)
LEFT OUTER JOIN TI_OFFERPRICE_O TI_OFFERPRICE ON
(CATENTRY.CATENTRY_ID=TI_OFFERPRICE.CATENTRY_ID)
LEFT OUTER JOIN TI_DPCATENTRY_O TI_DPCATENTRY ON
(CATENTRY.CATENTRY_ID=TI_DPCATENTRY.CATENTRY_ID)
LEFT OUTER JOIN TI_CATALOG_O TI_CATALOG ON (CATENTRY.CATENTRY_ID=TI_CATALOG.CATENTRY_ID)

LEFT OUTER JOIN XI_PRODUCT_INVENTORY_O XI_PRODUCT_INVENTORY ON (CATENTRY.CATENTRY_ID=
CAST(XI_PRODUCT_INVENTORY.CATENTRY_ID_PARENT AS BIGINT) )

LEFT OUTER JOIN TI_CASTB1_O_1 TI_CASTB1 ON (CATENTRY.CATENTRY_ID=TI_CASTB1.CATENTRY_ID)
LEFT OUTER JOIN TI_CAITB1_O_1 TI_CAITB1 ON (CATENTRY.CATENTRY_ID=TI_CAITB1.CATENTRY_ID)
LEFT OUTER JOIN TI_CAFTB1_O_1 TI_CAFTB1 ON (CATENTRY.CATENTRY_ID=TI_CAFTB1.CATENTRY_ID)
LEFT OUTER JOIN TI_ADSTB1_O_1 TI_ADSTB1 ON (CATENTRY.CATENTRY_ID=TI_ADSTB1.CATENTRY_ID)
LEFT OUTER JOIN TI_ADITB1_O_1 TI_ADITB1 ON (CATENTRY.CATENTRY_ID=TI_ADITB1.CATENTRY_ID)
LEFT OUTER JOIN TI_ADFTB1_O_1 TI_ADFTB1 ON (CATENTRY.CATENTRY_ID=TI_ADFTB1.CATENTRY_ID)
WHERE  CATENTRY.CATENTTYPE_ID='ProductBean'
```

iii. Add the child node (see Example 7-7).

Example 7-7 Child nodes

```
<field column="CATENTRY_ID" name="catentry_id" />
<field column="MEMBER_ID" name="member_id" />
<field column="CATENTTYPE_ID" name="catenttype_id_ntk_cs" />
```

```
<field column="PARTNUMBER" name="partNumber_ntk" />
<field column="MFPARTNUMBER" name="mfPartNumber_ntk" />
<field column="MFNAME" name="mfName" />
<field column="BUYABLE" name="buyable" />
<field column="STOREENT_ID" name="storeent_id" />
<field column="NAME" name="name"  />
<field column="SHORTDESCRIPTION" name="shortDescription" />
<field column="THUMBNAIL" name="thumbnail" />
<field column="FULLIMAGE" name="fullImage" />
<field column="KEYWORD" name="keyword" />
<field column="PUBLISHED" name="published" />
<field column="parentCatgroup_id_facet" splitBy=";" sourceColName="DPCATGROUP"/>
<field column="parentCatgroup_id_search" splitBy=";" sourceColName="APCATGROUP"/>
<field column="parentCatentry_id" splitBy=";" sourceColName="CATENTRY_PARENT"/>
<field column="catalog_id" splitBy=";" sourceColName="PARENT_CATALOG_ID"/>
<field column="productset_id" splitBy=";" sourceColName="PRODUCTSET"/>

<field column="PRICE_USD" name="price_USD"/>
<field column="PRICE_EUR" name="price_EUR"/>
<field column="PRICE_CAD" name="price_CAD"/>
<field column="PRICE_CNY" name="price_CNY"/>
<field column="PRICE_JPY" name="price_JPY"/>
<field column="PRICE_KRW" name="price_KRW"/>
<field column="PRICE_BRL" name="price_BRL"/>
<field column="PRICE_TWD" name="price_TWD"/>
<field column="PRICE_PLN" name="price_PLN"/>
<field column="PRICE_RON" name="price_RON"/>
<field column="PRICE_RUB" name="price_RUB"/>
<field column="PRICE_EGP" name="price_EGP"/>
<field column="PRICE_GBP" name="price_GBP"/>

<field column="XQUANTITY" name="xquantity"/>

<field column="CAS_F1ATTR"  clob="true"/>
<field column="cas_f1" splitBy=";" sourceColName="CAS_F1ATTR"/>
<field column="cas_f1_ntk_cs" splitBy=";" sourceColName="CAS_F1ATTR"/>
<!-- CAS_F2ATTR through CAS_F29ATTR removed -->
<!-- cas_f2 through cas_f29 removed -->
<!-- cas_f2_ntk_cs through cas_f29_ntk_cs removed -->
<field column="CAS_F30ATTR"  clob="true"/>
<field column="cas_f30" splitBy=";" sourceColName="CAS_F30ATTR"/>
<field column="cas_f30_ntk_cs" splitBy=";" sourceColName="CAS_F30ATTR"/>

<field column="cai_f1" splitBy=";" sourceColName="CAI_F1"/>
<!-- caf_i2 through cai_f9 removed -->
<field column="cai_f10" splitBy=";" sourceColName="CAI_F10"/>

<field column="caf_f1" splitBy=";" sourceColName="CAF_F1"/>
<!-- caf_f2 through caf_f9 removed -->
<field column="caf_f10" splitBy=";" sourceColName="CAF_F10"/>

<field column="ADS_F1ATTR"  clob="true"/>
<field column="ads_f1" splitBy=";" sourceColName="ADS_F1ATTR"/>
<field column="ads_f1_ntk_cs" splitBy=";" sourceColName="ADS_F1ATTR"/>
<!-- ADS_F2ATTR through ADS_F29ATTR removed -->
```

```
<!-- ads_f2 through ads_f29 removed -->
<!-- ads_f2_ntk_cs through ads_f29_ntk_cs removed -->
<field column="ADS_F30ATTR"  clob="true"/>
<field column="ads_f30" splitBy=";" sourceColName="ADS_F30ATTR"/>
<field column="ads_f30_ntk_cs" splitBy=";" sourceColName="ADS_F30ATTR"/>

<field column="adi_f1" splitBy=";" sourceColName="ADI_F1"/>
<!-- adi_f2 through adi_f9 removed -->
<field column="adi_f10" splitBy=";" sourceColName="ADI_F10"/>

<field column="adf_f1" splitBy=";" sourceColName="ADF_F1"/>
<!-- adf_f2 through adi_f9 removed -->
<field column="adf_f10" splitBy=";" sourceColName="ADF_F10"/>
```

h. Save and close the file.

i. Restart the WebSphere Commerce Developer Edition V7 test server.

j. Build the search index. See 7.2.4, “Building the search index” on page 185.

k. Verify the data:

```
http://localhost/solr/MC_10051_CatalogEntry_en_US/select?q=catenttype_id_ntk
_cs:ItemBean&fl=*
```

Troubleshooting: If your XI_<*TABLENAME*> is missing data in a column, check that you have all of your column/column mappings.

3. Add our quantity filter query to all queries:

   a. Open the Notepad.

   b. **Click File** → **Open** to open this file:

   ```
   c:\IBM\WCDE_ENT70\search\solr\home\MC_10051\en_US\CatalogEntry\conf\solrconf
   ig.xml
   ```

   c. Find the //config/requestHandler[@name='standard'] node.

   d. Add the quantity filter query (see Example 7-8).

   e. Restart the WebSphere Commerce Developer Edition V7 test server.

   f. Verify the quantity filter:

   ```
   http://localhost/solr/MC_10051_CatalogEntry_en_US/select?q=*&fl=*
   ```

Example 7-8 solrconfig.xml snippet

```
<config>
   <!-- content removed for clarity -->
   <!-- WebSphere Commerce default search request handler -->
   <requestHandler name="standard" class="solr.StandardRequestHandler" default="true">
      <!-- default values for query parameters -->
      <lst name="defaults">
         <str name="echoParams">explicit</str>
         <!--
         <int name="rows">10</int>
         <str name="fl">*</str>
         <str name="version">2.1</str>
         -->
      </lst>
```

```
      <!-- Filter all queries to ensure returned catalog entries have quantity -->
      <lst name="appends">
         <str name="fq">xquantity:[1 TO *]</str>
      </lst>

      <arr name="last-components">
         <str>wc_spellcheck</str>
      </arr>
   </requestHandler>
   <!-- content removed for clarity -->
</config>
```

7.3.2 Index maintenance

Automatic re-indexing is an event-driven process in WebSphere Commerce V7.0. When a message is received to mutate the base schema, ChangeCatalogEntry events are raised and handled by WebSphere Commerce V7.0. These handlers parse the event and mutate TI_DELTA_CATENTRY. Then, when a CategoryNavigationView message is received, it raises a SynchronizeIndexEvent if the table is not empty. Neither case raises a ChangeCatalogEntry event; therefore the TI_DELTA_CATENTRY table is not mutated and we must use the following re-indexing strategies.

The maintenance for our customized index introduces two separate index update cases:

- The inventory update feed is received and loaded into the WebSphere Commerce V7.0 database. We run a full re-index after each inventory feed load to handle this case.
- An inventory update is caused by a storefront order. We set up a scheduled job, RBInventoryDeltaUpdateIndexCmd, which is a sample exercise, that generates an XML message for all catalog entries that have INVENTORY.QUANTITY = 0. Then, we HTTP/POST the message to the WebSphere Commerce V7.0 search server.

Follow these steps:

1. Schedule RBInventoryDeltaIndexCmd to run at a certain frequency, *f1*, that will generate an `inventory_update_message.xml` (see Example 7-9).

   Example 7-9 inventory_update_message.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<update>
   <add>
      <doc>
         <field name="buyable">1</field>
         <field name="catalog_id">10001</field>
         <field name="catentry_id">10246</field>
         <field name="catenttype_id_ntk_cs">ProductBean</field>
         <field
name="fullImage">images/catalog/apparel/apparel_160x160/IMG_0054_e.jpg</field>
         <field name="member_id">7000000000000000002</field>
         <field name="mfName">MapleWear</field>
         <field name="mfName_ntk">MapleWear</field>
         <field name="mfName_ntk_cs">MapleWear</field>
         <field name="name">Bodysuit</field>
         <field name="parentCatgroup_id_facet">10001_10031</field>
<!-- handle arrays by flattening and repeating the field name element -->
```

```
        <field name="parentCatgroup_id_search">10001_10031</field>
        <field name="parentCatgroup_id_search">10001_10029</field>
        <field name="partNumber_ntk">MW-0054</field>
        <field name="price_USD">6.99</field>
        <field name="published">1</field>
        <field name="shortDescription">Short-sleeved bodysuit.</field>
        <field name="storeent_id">10001</field>
        <field
name="thumbnail">images/catalog/apparel/apparel_70x70/IMG_0054_e.jpg</field>
        <field name="xquantity">30</field>
     </doc>
     <!-- you can have multiple <doc> ... </doc> in one add element -->
  </add>
  <!-- inside the update element you can have add's and delete's
  <delete>
     <id>10245</id>
  </delete>
</update>
```

2. HTTP/POST the `inventory_update_message.xml` to the search server using `post.sh` (see Example 7-10):

```
redbooks@linux-8sjm:~>./post.sh inventory_update_message.xml
```

Example 7-10 post.sh

```
#!/usr/bin/sh
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

FILES=$*
URL=http://linux-8sjm.site:3737/solr/MC_10001_CatalogEntry_en_US/update

echo $URL

for f in $FILES; do
  echo Posting file $f to $URL
  curl $URL --data-binary @$f -H 'Content-type:application/xml'
  echo
done

#send the commit command to make sure all the changes are flushed and visible
curl $URL --data-binary '<commit/>' -H 'Content-type:application/xml'
echo "complete"
```

Solr XML indexing:

- Document the actions that can be performed: add, delete, commit, rollback, and optimize.
- Document that the updates are composed of a document delete followed by a document add.
- No changes to the index are visible until the commit action is performed.
- For more information, go to http://wiki.apache.org/solr/UpdateXmlMessages.

7.4 Building catalog indexes from XML files

Solr V1.3 provides an interface to build indexes from XML files. The challenge is how to resolve the WebSphere Commerce V7.0 metadata information. WebSphere Commerce depends on the metadata in the index, for example, store_id, catalog_id, and productset_id, to filter out the entitled products based on the user's business context. This metadata is the internal ID of the objects. Most metadata is generated by the WebSphere Commerce key manager function. So, if the XML is built from a non-WebSphere Commerce authoring system, it can be challenging to resolve the metadata.

Another consideration for building the index from the XML files is that part of the catalog data is still required to be loaded in the WebSphere Commerce database to allow WebSphere Commerce components, such as order and promotion, to function properly.

In general, when the client manages the product information in a non-WebSphere Commerce authoring system, we suggest that the client first load the product data to the WebSphere Commerce database and then build the WebSphere Commerce search index from the WebSphere Commerce database. Use these high-level steps:

1. Identify the products that need to be extracted from the non-WebSphere Commerce authoring system.
2. Extract the products from the authoring system to files (use comma-separated value (CSV) files).
3. Load the CSV files into the WebSphere Commerce database with the WebSphere Commerce data load utility. The utility resolves and generates the WebSphere Commerce internal IDs, including the metadata internal IDs.
4. Verify the data in the WebSphere Commerce system.
5. Prepare the index building data with the WebSphere Commerce index preprocess utility (see 7.2.3, "Preprocessing the index data" on page 184).
6. Build the search index with the WebSphere Commerce index building utility (see 7.2.4, "Building the search index" on page 185).

In this process, the WebSphere Commerce dataload is used to generate the internal IDs and to load the data into the WebSphere Commerce database.

Often, clients want to load external data (for example, the product rating and reviewing information) into the search index, but they do not necessarily need to load the data into the WebSphere Commerce database. In these cases, using the XML file to load the data into the catalog and unstructured content indexes can be appropriate, provided that the metadata is also loaded into the index properly.

7.5 Reference

In this section, we document the WebSphere Commerce V7.0 index information.

7.5.1 WebSphere Commerce V7.0 search metadata

Table 7-1 lists the fields that comprise the CatalogEntry index and the database tables that populate the fields.

Table 7-1 Search metadata

Number	Index metadata	Comment
0	catentry_id	The CATENTRY_ID column of the CATENTRY table.
1	member_id	This is the owner's ID of the product. The MEMBER_ID column of the CATENTRY table.
2	catenttype_id_ntk_cs	The CATENTTYPE_ID column of the CATENTRY table.
3	partNumber_ntk	The PARTNUMBER column of the CATENTRY table.
4	mfPartNumber_ntk	The MFPARTNUMBER column of the CATENTRY table.
5	mfName	The MFNAME column of the CATENTRY table.
6	buyable	The BUYABLE column of the CATENTRY table.
7	storeent_id	This is the store to which the product belongs. For example, if the product belongs to catalog asset store 10, only store 10 is indexed. When e-site stores inherit the product, the index does not need to be updated. The STOREENT_ID column of STORECENT table for the catentry_id.
8	name	The NAME column of the CATENTDESC table for the catentry_id and the specified language_id.
9	shortDescription	The SHORTDESCRIPTION column of the CATENTDESC table for the catentry_id and the specified language_id.
10	thumbnail	The THUMBNAIL column of the CATENTDESC table for the catentry_id and the specified language_id.
11	fullImage	The FULLIMAGE column of the CATENTDESC table for the catentry_id and the specified language_id.
12	keyword	The KEYWORD column of the CATENTDESC table for the catentry_id and the specified language_id.
13	published	The PUBLISHED column of the CATENTDESC table for the catentry_id and the specified language_id.

Number	Index metadata	Comment
14	parentCatgroup_id_facet	This is a multiple value field. Each value has two parts connected by "_", the first part is the CATALOG_ID, and the second part is the CATGROUP_ID, to which the catalog entry directly belongs, for example, 10101_10200. The sql to get the data is: SELECT CATGPENREL.CATENTRY_ID CATENTRY_ID, rtrim(cast(catgpenrel.catalog_id as char(16))) ||'_'|| rtrim(cast(CATGPENREL.CATGROUP_ID as char(16))) CATGROUP_ID FROM CATGPENREL, STORECENT, STORECAT WHERE CATGPENREL.CATENTRY_ID=STORECENT.CATENTRY_ID AND (STORECENT.STOREENT_ID IN (SELECT STOREREL.STORE_ID FROM STOREREL WHERE STOREREL.STRELTYP_ID=-4 AND STOREREL.RELATEDSTORE_ID=STORECAT.STOREENT_ID) OR STORECENT.STOREENT_ID = STORECAT.STOREENT_ID) AND STORECAT.CATALOG_ID=CATGPENREL.CATALOG_ID AND CATGPENREL.CATENTRY_ID=*XXXXX* Where *XXXXX* is the `catentry_id`.
15	parentCatgroup_id_search	This is the product's parent group list (including all levels in the category hierarchy tree). From the CATGPENREL table, find the direct parent catgroup_id, after, based on the catgroup_id, using the CATGRPREL table to find all its ancestors for the catalog entry.
16	parentCatentry_id	This is the catentry's parent catentry, for example, an item's parent product. It is easy to find the catentry's parent catentry from the CATENTREL table. The sql to get the data is: SELECT CATENTRY_ID_CHILD, CATENTRY_ID_PARENT FROM CATENTREL R WHERE R.CATENTRY_ID_CHILD = *XXXXX* Where *XXXXX* is the catentry_id.
17	catalog_id	This is the catalog to which the product belongs. The sql to get the data is: SELECT DISTINCT R.CATENTRY_ID CATENTRY_ID, R.CATALOG_ID CATALOG_ID FROM CATGPENREL R WHERE R.CATENTRY_ID = *XXXXX* Where *XXXXX* is the catentry_id.
18	productset_id	This is the product set to which the product belongs. It is easy to find the product sets from the PSSETCEREL table. The sql to get the data is: SELECT PRSETCEREL.CATENTRY_ID, PRODUCTSET.PRODUCTSET_ID FROM PRSETCEREL, PRODUCTSET WHERE PRODUCTSET.PRODUCTSET_ID = PRSETCEREL.PRODUCTSET_ID AND PRODUCTSET.MARKFORDELETE = 0 AND PRSETCEREL.CATENTRY_ID=*XXXXX* Where *XXXXX* is the catentry_id.

Number	Index metadata	Comment
19	price_USD	The offer price in USD (US dollars). The offer price for a catalog entry, which is not a “BundleBean”, is the PRICE column in the OFFERPRICE table for the offer, which has the highest PRECEDENCE, and the MINIMUMQUANTITY column has either 1 or 0, or is null. The sql to get the offer_id is: 1. select o.offer_id offer_id, o.catentry_id catentry_id from offer o, TRADEPOSCN T, CATGRPTPC C, STORECENT S, (select max(PRECEDENCE) as maxpre, offer_id from offer where (MINIMUMQUANTITY IN (1, 0) OR MINIMUMQUANTITY IS NULL) group by offer_id) o2 where O.TRADEPOSCN_ID=T.TRADEPOSCN_ID and T.TRADEPOSCN_ID=C.TRADEPOSCN_ID and C.STORE_ID=S.STOREENT_ID and o.catentry_id=S.catentry_id and o.catentry_id=*XXXXX* and o.offer_id=o2.offer_id order by offer_id Where XXXXX is the catentry_id. The sql to get the offer price is: 2. select o.catentry_id catentry_id, p.currency currency, p.price price from offerprice p, where p.offer_id = *YYYYY* AND p.currency in ('USD') AND CE.CATENTTYPE_ID <> 'BundleBean' AND CE.CATENTRY_ID=*XXXXX* Where *XXXXX* is the catentry_id and *YYYYY* is the offer_id from the getting offer_id sql shown before this sql. For the catalog entry that has the type of “BundleBean” The price is the sum of all the catalog entries that belong to the bundle. Using the CATENTREL, where the CATENTREL.CATRELTYPE_ID='BUNDLE_COMPONENT' and CATENTREL.CATENTRY_ID_PARENT=*XXXXX* Where *XXXXX* is the catentry_id.
20	price_EUR	The offer price for EUR (Euro). How to get the price is the same as the price_USD except the currency is "EUR".
21	price_CAD	The offer price for CAD (Canadian Dollar). How to get the price is the same as the price_USD except the currency is "CAD".
22	price_CNY	The offer price for CNY (Chinese Yen). How to get the price is the same as the price_USD except the currency is "CNY".
23	price_JPY	The offer price for JPY (Japanese Yen). How to get the price is the same as the price_USD except the currency is "JPY".
24	price_KRW	The offer price for KRW (Korean Won). How to get the price is the same as the price_USD except the currency is "KRW".
25	price_BRL	The offer price for BRL (Brazilian Real). How to get the price is the same as the price_USD except the currency is "BRL".
26	price_TWD	The offer price for TWD (New Taiwan Dollar). How to get the price is the same as the price_USD except the currency is "TWD".
27	price_PLN	The offer price for PLN (Polish Zloty). How to get the price is the same as the price_USD except the currency is "PLN".
28	price_RON	The offer price for RON (Romanian Lei). How to get the price is the same as the price_USD except the currency is "RON".
29	price_RUB	The offer price for RUB (Russian Ruble). How to get the price is the same as the price_USD except the currency is "RUB".

Number	Index metadata	Comment
30	price_EGP	The offer price for EGP (Egyptian Pound). How to get the price is the same as the price_USD except the currency is "EGP".
31	price_GBP	The offer price for GBP (British Pound). How to get the price is the same as the price_USD except the currency is "GBP".
32-61	ads_f1 to ads_f30	The attribute values for the string type of classic attributes. The classic attribute mapping information is stored in the ATTRDICTSRCHCONF table. You need to populate the correct information for the ATTR_ID column into this table for the attribute in your master catalog ID. You will know which solr index field is for which attribute. The product has all the values of its SKUs.
62-71	adi_f1 to adi_f10	The attribute values for the integer type of classic attributes. The classic attribute mapping information is stored in the ATTRDICTSRCHCONF table. You need to populate the correct information for the ATTR_ID column into this table for the attribute in your master catalog ID. You will know which solr index field is for which attribute. The product has all the values of its SKUs.
72-81	adf_f1 to adf_f10	The attribute values for the integer type of classic attributes. The classic attribute mapping information is stored in ATTRDICTSRCHCONF table. You need to populate the correct information for the ATTR_ID column into this table for the attribute in your master catalog ID. You will know which solr index field is for which attribute. The product has all the values of its SKUs.
82-111	cas_f1 to cas_f30	The attribute values for the string type of classic attributes. The classic attribute mapping information is stored in the CLSATTRSRCHCONF table. You need to populate the correct information for the ATTRNAME column into this table for the attribute in your master catalog ID and language. You will know which solr index field is for which attribute. The product has all the values of its SKUs.
112-121	cai_f1 to cai_f10	The attribute values for the integer type of classic attributes. The classic attribute mapping information is stored in the CLSATTRSRCHCONF table. You need to populate the correct information for the ATTRNAME column into this table for the attribute in your master catalog ID and language. You will know which solr index field is for which attribute. The product has all the values of its SKUs.
122-131	caf_f1 to caf_f10	The attribute values for the float type of classic attributes. The classic attribute mapping information is stored in the CLSATTRSRCHCONF table. You need to populate the correct information for the ATTRNAME column into this table for the attribute in your master catalog ID and language. You will know which solr index field is for which attribute. The product has all the values of its SKUs.

8

Storefront

This chapter describes how to enable the search-based navigation for a storefront and discusses all the new features that are available as part of the predefined search when enabled. We implemented all the scenarios in this chapter in the MadisonsFEP2 business-to-consumer (B2C) store.

Names: The following company names appearing in this publication are fictitious. We use these names for instructional purposes only.

We use these search terms in this chapter:

- Facet: A property common to a set of search results. It can be static or dynamic, such as a brand name, price, or so on.
- Unstructured content: Content that is associated with a product, such as an instruction manual.
- Landing page: A predefined destination web page in the storefront.
- Stop word: Words that are filtered out prior to submitting the search, for example, “the” or “this”.

The storefront, by default, enables shoppers to navigate the catalog in the following ways:

- Quick search (by keyword): A search field with an automatic suggestion feature, which is integrated into the store’s header at the top of the page. This method allows shoppers to perform quick searches from any store page within the site.
- Taxonomy navigation (by category): A category-based navigation technique, which is driven by the header’s top navigation menu. This method is only effective when the shopper knows exactly in which category the desired products are located.
- Faceted navigation (by facet): This method is used for accessing product information that is represented using a faceted classification. This technique allows shoppers to explore by filtering information. A faceted classification system allows the assignment of multiple classifications to an object, enabling the classifications to be ordered in multiple ways, rather than in a single, predetermined, and taxonomic order. Each facet typically corresponds to the possible values of a property that is common to a set of objects.

- Advanced search (by keyword): This method is a variation of the quick search that allows you to specify more detailed searching criteria. This method is useful when the shopper knows a certain range of criteria or conditions to use to narrow the scope of the search results.

8.1 Enabling search-based navigation

Before using the enhanced search features that are provided in Feature Pack 2, we need to make sure that the store's master catalog is indexed for WebSphere Commerce search and the search server and that index structure is deployed and built.

To enable catalog-based navigation, perform the following steps:

1. Log on to the Management Center.
2. Select **Store Management** from the menu, as shown in Figure 8-1.

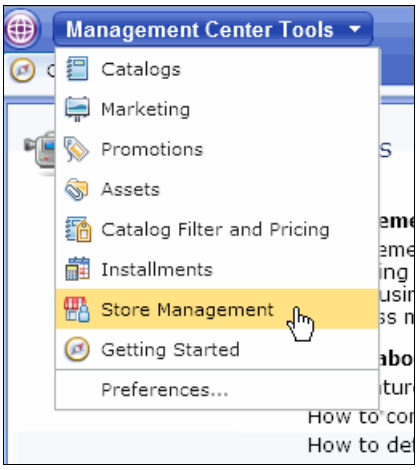


Figure 8-1 Opening the Store Management tool

3. Click **Stores** in the explorer view, as shown in Figure 8-2.

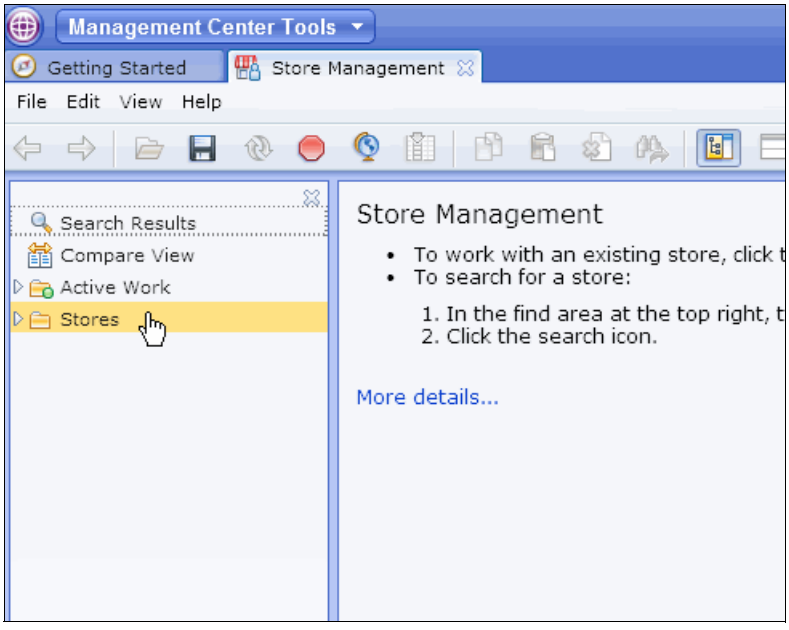


Figure 8-2 Selecting Stores list view

4. Right-click the store and click **Open**, as shown in Figure 8-3.

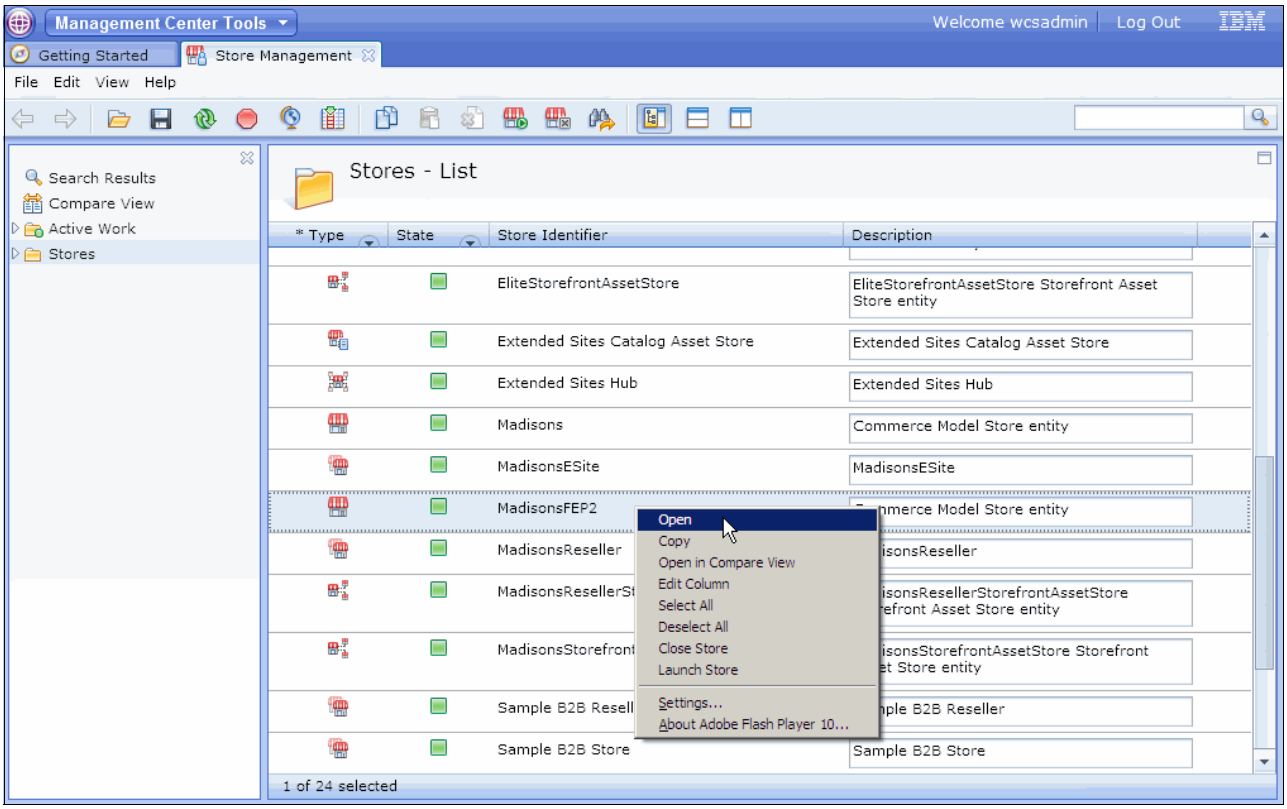


Figure 8-3 Opening MadisonsFEP2 store

5. Select the **Catalog** tab.

6. Select **Search-based navigation**, as shown in Figure 8-4.

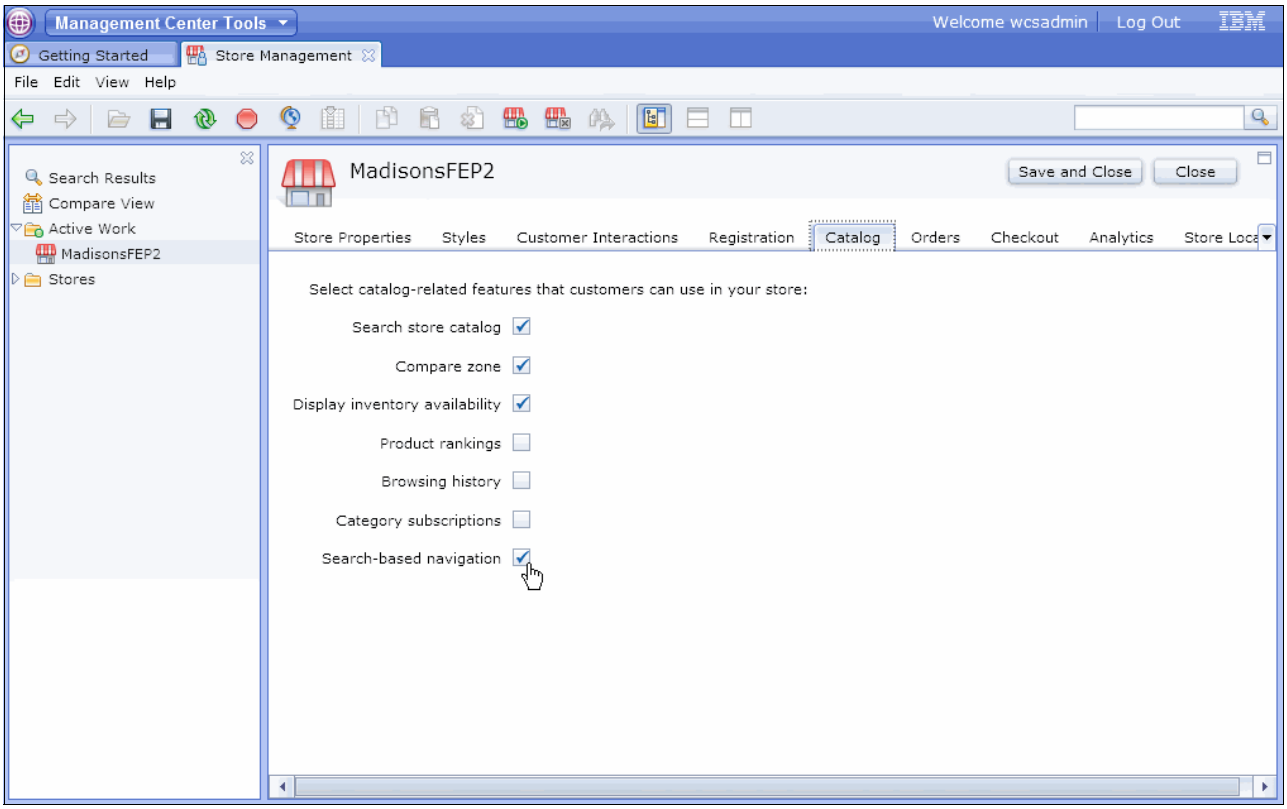


Figure 8-4 Enabling Search-based navigation

The storefront now has the following features enabled:

- Full-text search
- Auto suggest
- Spell correction
- Phrase search
- Wild cards
- Interactive breadcrumb trail with facet selected
- Search term highlighting
- Facets display
- Search unstructured content
- Multiple views (grid and list view)
- Displayed price can be indexed or calculated
- Configurable sort options
- Results can be dragged and dropped
- Results can be bookmarked

Most of the search scenarios in this chapter are performed by entering the search term in the quick search tool in the header. The breadcrumb trail in the Search Results page displays the input search term that was used.

8.2 Full-text search

Shoppers can enter a full word or a set of words in the quick search text box. For example, if the shopper enters `coffee maker`, the search is performed on the full text, as it was entered, and the results are displayed, as shown in Figure 8-5.

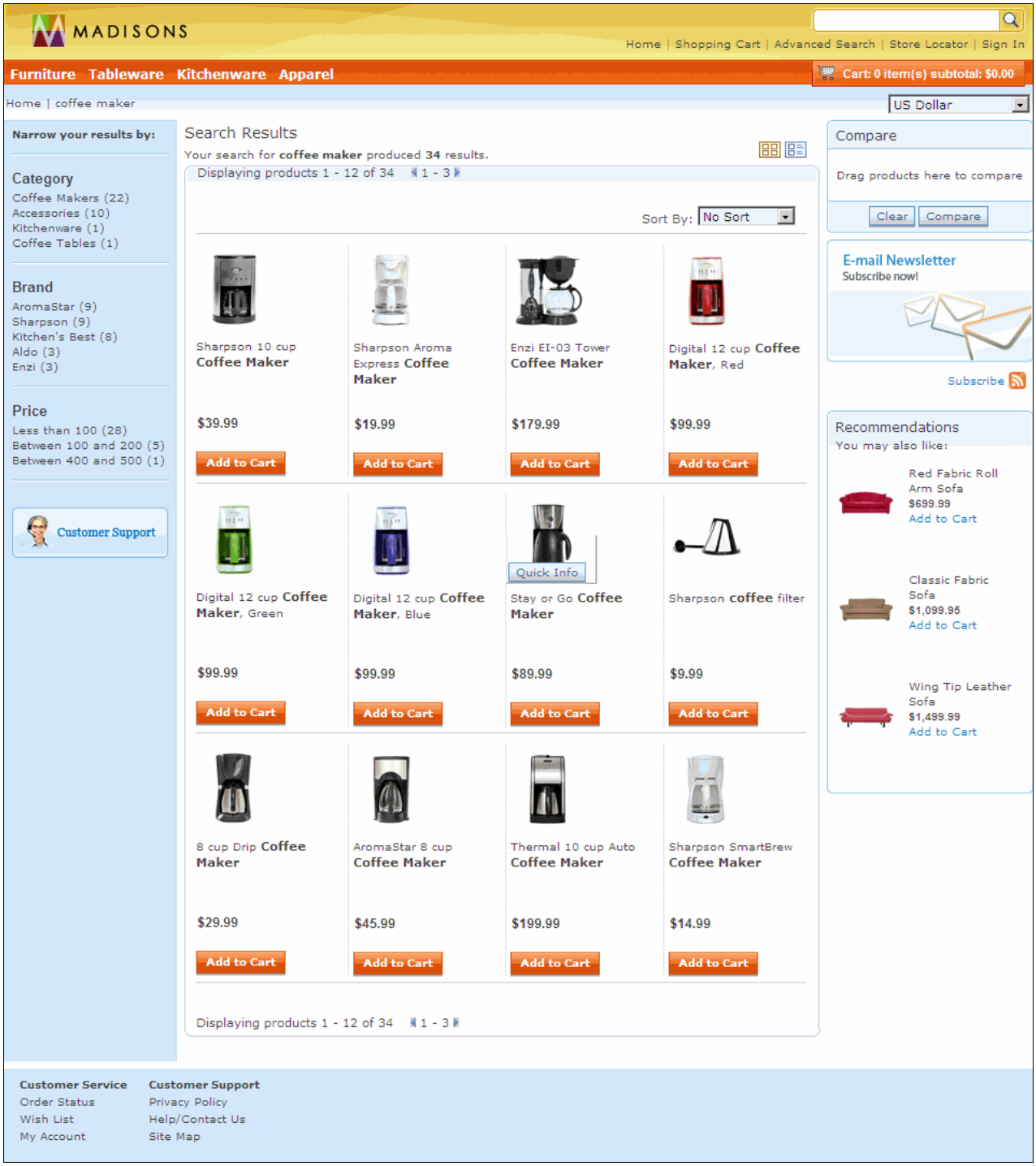


Figure 8-5 Search results for coffee maker

8.3 Auto suggest

The storefront provides shoppers with type-ahead functionality. Suggested keyword matches are automatically displayed and selectable underneath the search field when entering search terms in the store.

For example, when the shopper starts typing `so` into the quick search text box, the system shows a list of suggested terms, as shown in Figure 8-6.

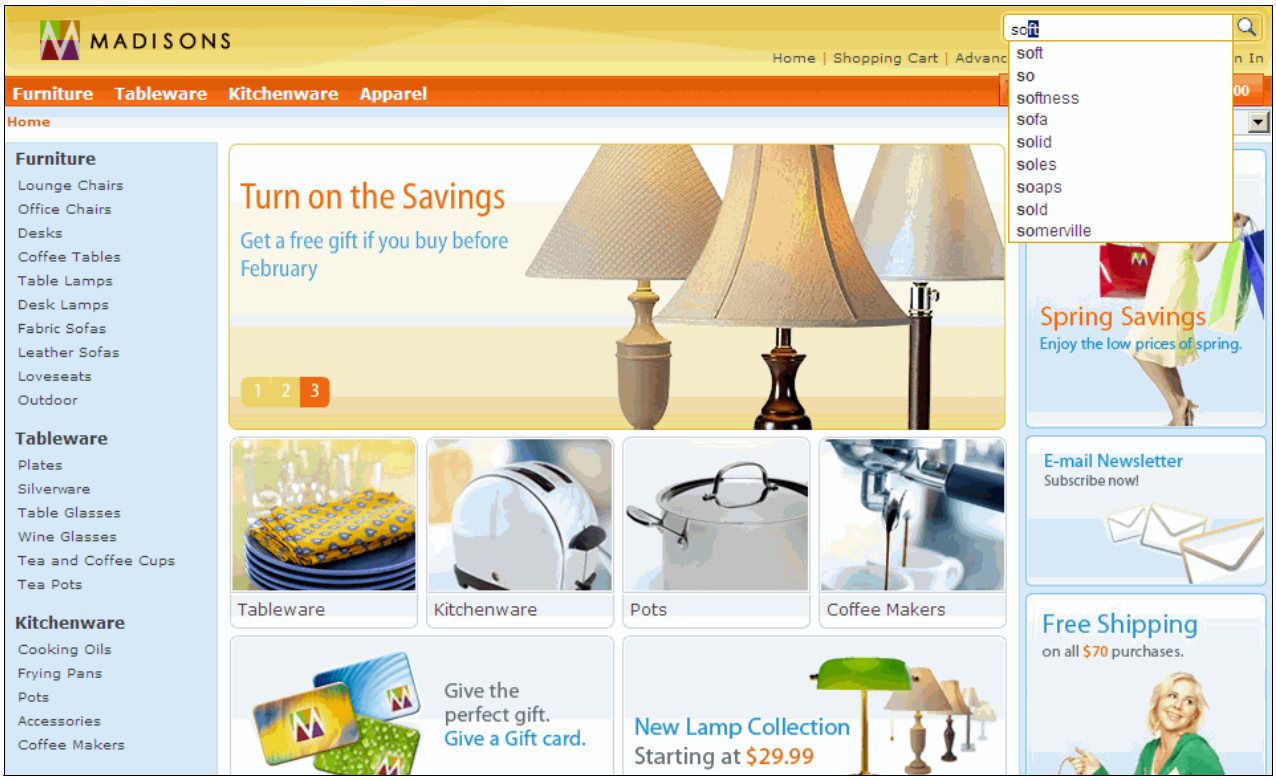


Figure 8-6 Auto suggest results for "so"

8.4 Spelling correction and search term suggestion

If the shopper enters a search term that is not found in the index, the system automatically runs the spell-checker at run time and suggests the nearest search term. It displays the results for the first spelling match that is returned. For example, if the shopper types `soaf` instead of `soap`, the system suggests the words: soaps, soft, star, sofa, and seat.

The system automatically displays the results for the first match, which is `soaps`, as shown in Figure 8-7.

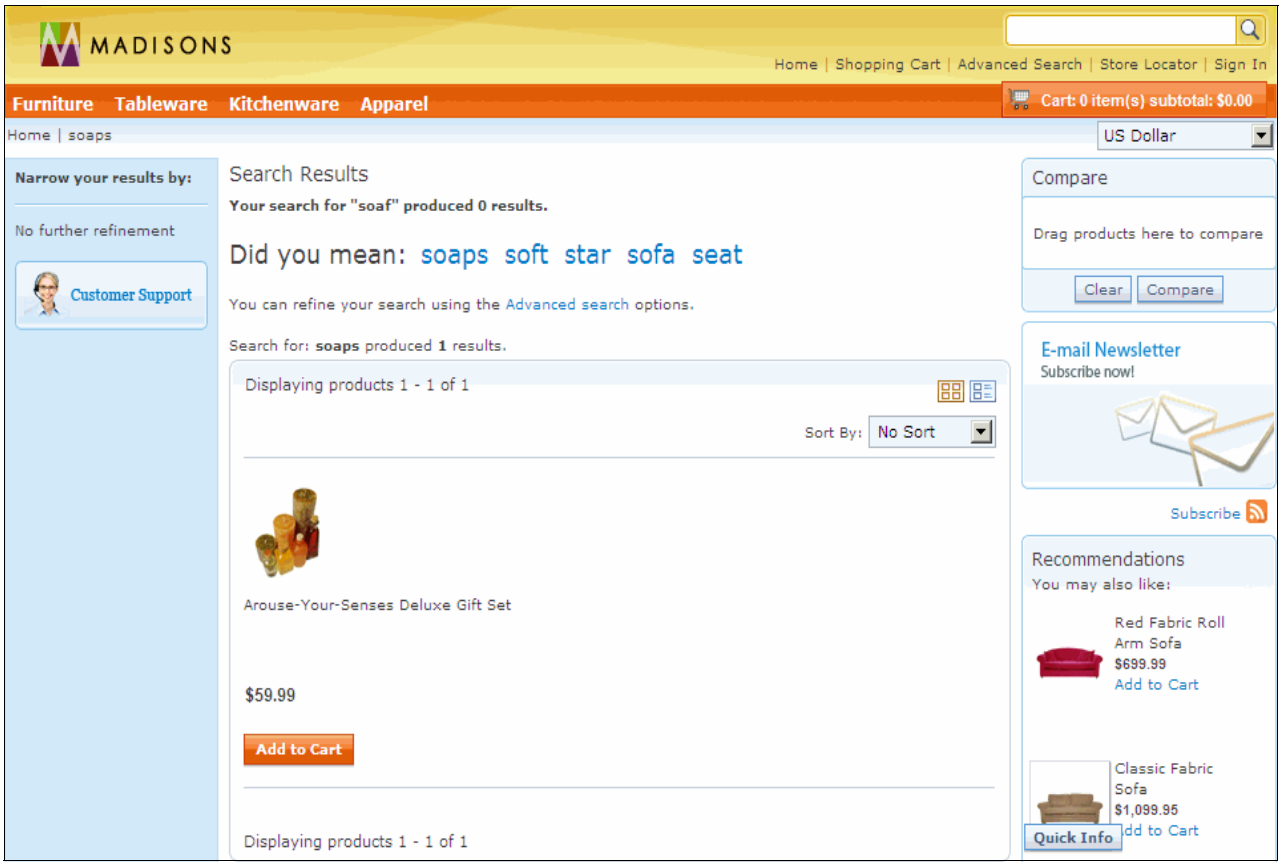


Figure 8-7 Search term suggestions in the Search Results page for "soaf"

The variable *limit* under the spellcheck node in `wc-search.xml` allows the configuration of the maximum number of suggestions to be returned and displayed in `Did you mean`. The variable *limit* is assigned a default value of 5, as shown in Example 8-1.

Example 8-1 Search term configuration

```
<_config:spellcheck>
<_config:param name="limit" value="5"/>
</_config:spellcheck>
```

8.5 Phrase search

Shoppers can enter a phrase in the quick search text box and the system returns results with products that have this phrase in the description. The system removes the *stop* words, which are defined in `stopword.txt`.

For example, if shoppers search for `This sofa offers`, the system removes the stop word `this` and displays the results for the phrase `sofa offers`, as shown in Figure 8-8.

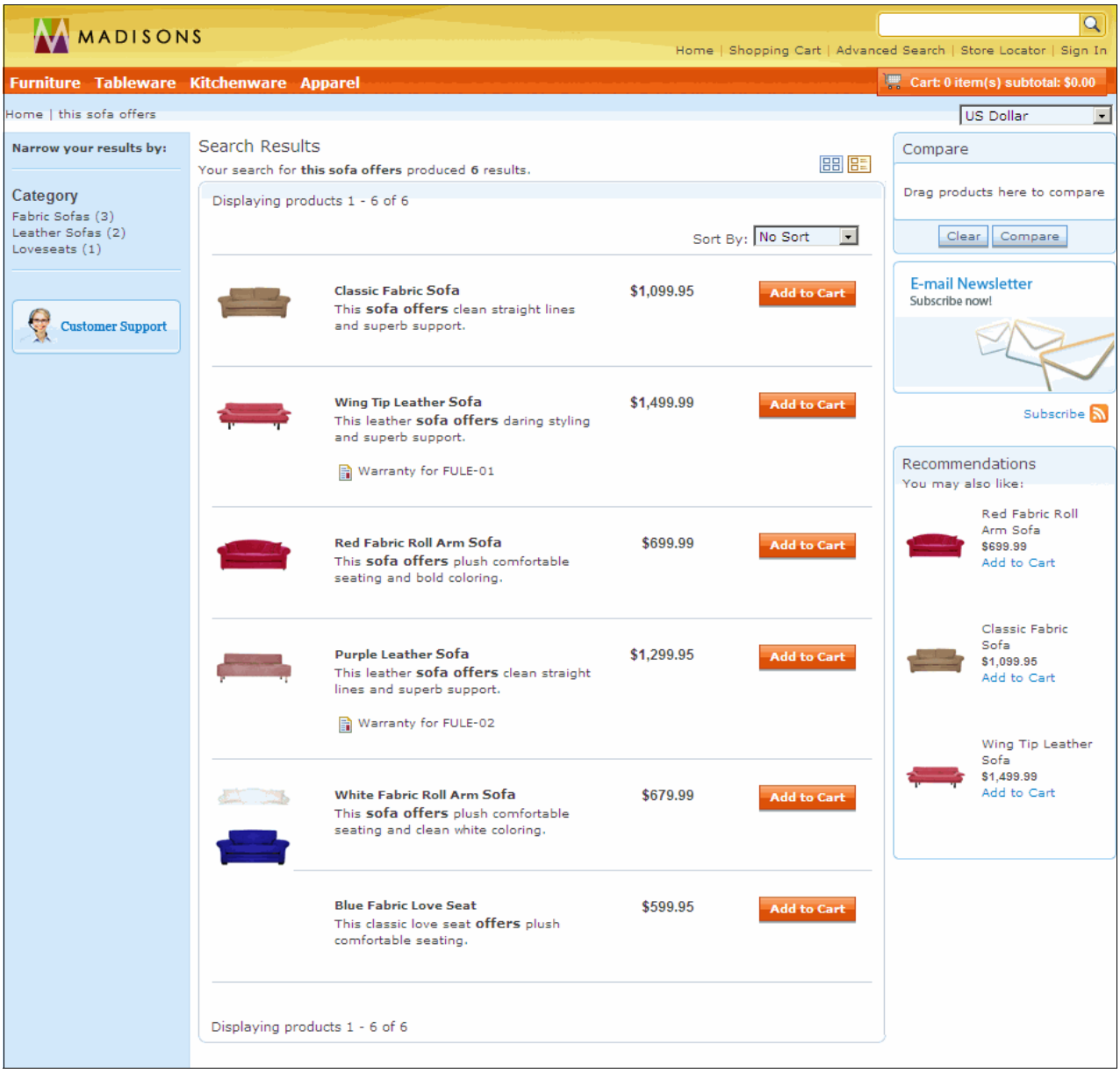


Figure 8-8 Search results for sofa offers

8.6 Wild cards

Search terms can contain a wildcard character, an asterisk (*), to increase the scope of the search results. The wildcard indicates that all products containing the surrounding partial search term must be returned in the search results. Wild card searching can help more advanced search users quickly find and display large amounts of search results.

For example, submitting a search for `coff*` returns search results for all items that contain the term `coff`, including any terms following it, such as `coffee` or `coffee maker`, as shown in Figure 8-9.

When compared with the sample search results for automatic search term suggestions and spelling correction, instead of producing no search results using the search term `coff`, wild card searching produces several search results simply by adding the asterisk to the search term.

Wild card searching is enabled by default, but if necessary, you can disable it for runtime performance or security reasons:

- A prohibited words list stops the search request from further searching. You configure the prohibited words list in the `wc-component.xml` file. For example, when a shopper searches for * by default, the resulting page is routed to the Prohibited Characters store page.
- The following configuration is the default:

```
<_config:property name="StopPatterns"
value="\*,~,\?,&apos;&apos;,&quot;&quot;,.*\\.*,.*/.*,.*\|.*" />
```

- You can update the configuration by using the regular expression format.

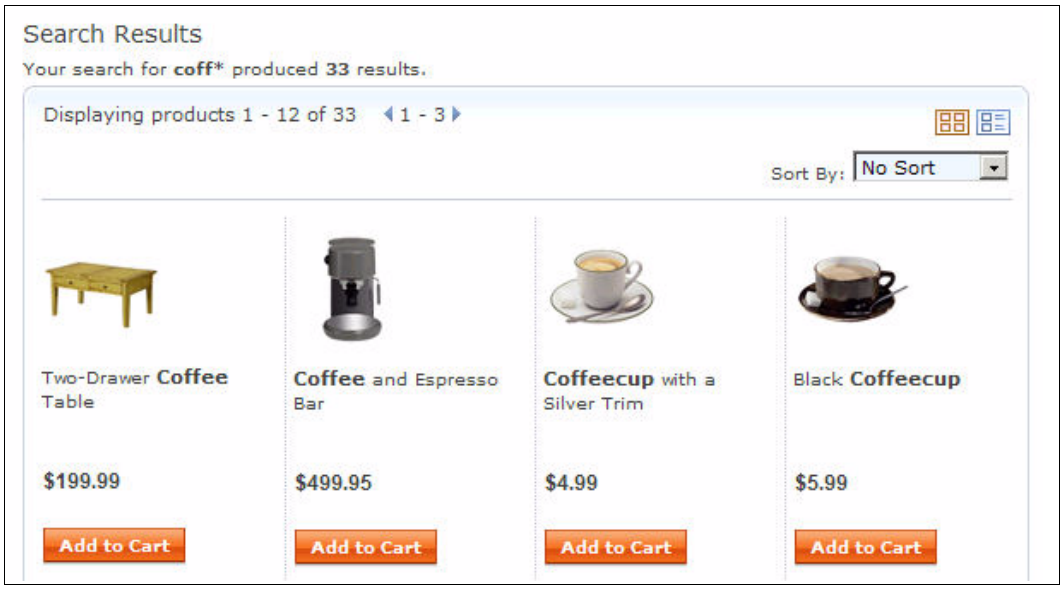


*Figure 8-9 Search results for coff**

The prohibit function is used together with the following configuration, which is also in the `wc-component.xml` file:

- <!-- ; splits sections. Each section's related fields are separated by , -->
- <_config:property name="RequestFields" value="searchTerm,filterTerm,manufacturer,minPrice,maxPrice,categoryId" />
- The "searchTerm,filterTerm,manufacturer,minPrice,maxPrice,categoryId" comes from the JavaServer Pages (JSP) field names, as marked in Figure 8-10 on page 215.

The meaning of the string is to concatenate all the values in these fields into one input string(StringA), and StringA will be matched against the prohibit list parts. If the concatenate StringA matches the prohibit string like "*", it will go to the Prohibit Error page.

This function is used for both advanced search and basic search.

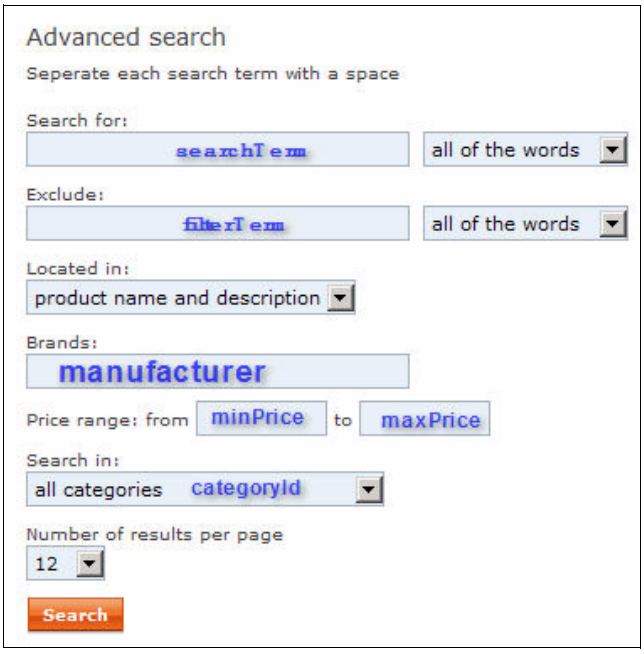


Figure 8-10 Advanced search

8.7 Facets display

Shoppers can find products easily in the storefront using faceted navigation. Facets group similar items and attributes together in the left section in the Search Results page to target meaningful product results. The groups and shared attributes are displayed with the number of products contained in each facet. Facets help filter the search results and are displayed as links in the left section of the Search Results page, such as brands, prices, and features.

Facets: All content that is displayed in facets must be indexed in the WebSphere Commerce search. For example, offer prices for separate currencies must be indexed to enable price facets and sorting.

Brands are not translated, by default, in search facets. Standard offer prices are indexed by default. That is, price facets are based on the standard offer price, where any adjustments from price rules or contract prices are not reflected on the price facets.

For example, when the shopper searches for `coffee`, the system returns 31 products and displays the facets, category, brand, and price, in the left section of the Search Results window, as shown in Figure 8-11.

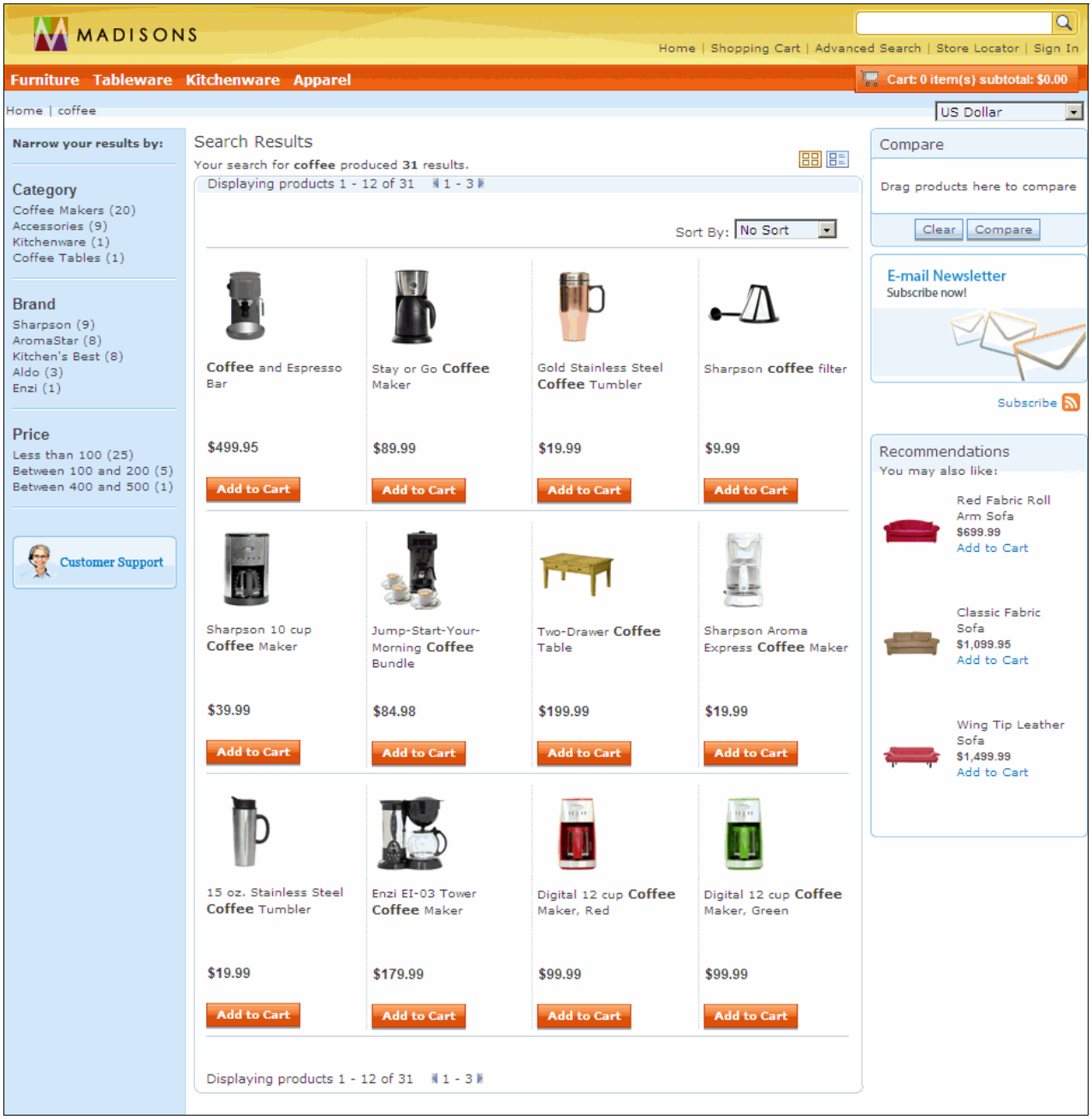


Figure 8-11 Facets shown in left section of the Search Results window

Selecting the facet **Less than 100** filters the results by price and it returns 25 products. It also displays the available facets, category and brand, as shown in Figure 8-12.

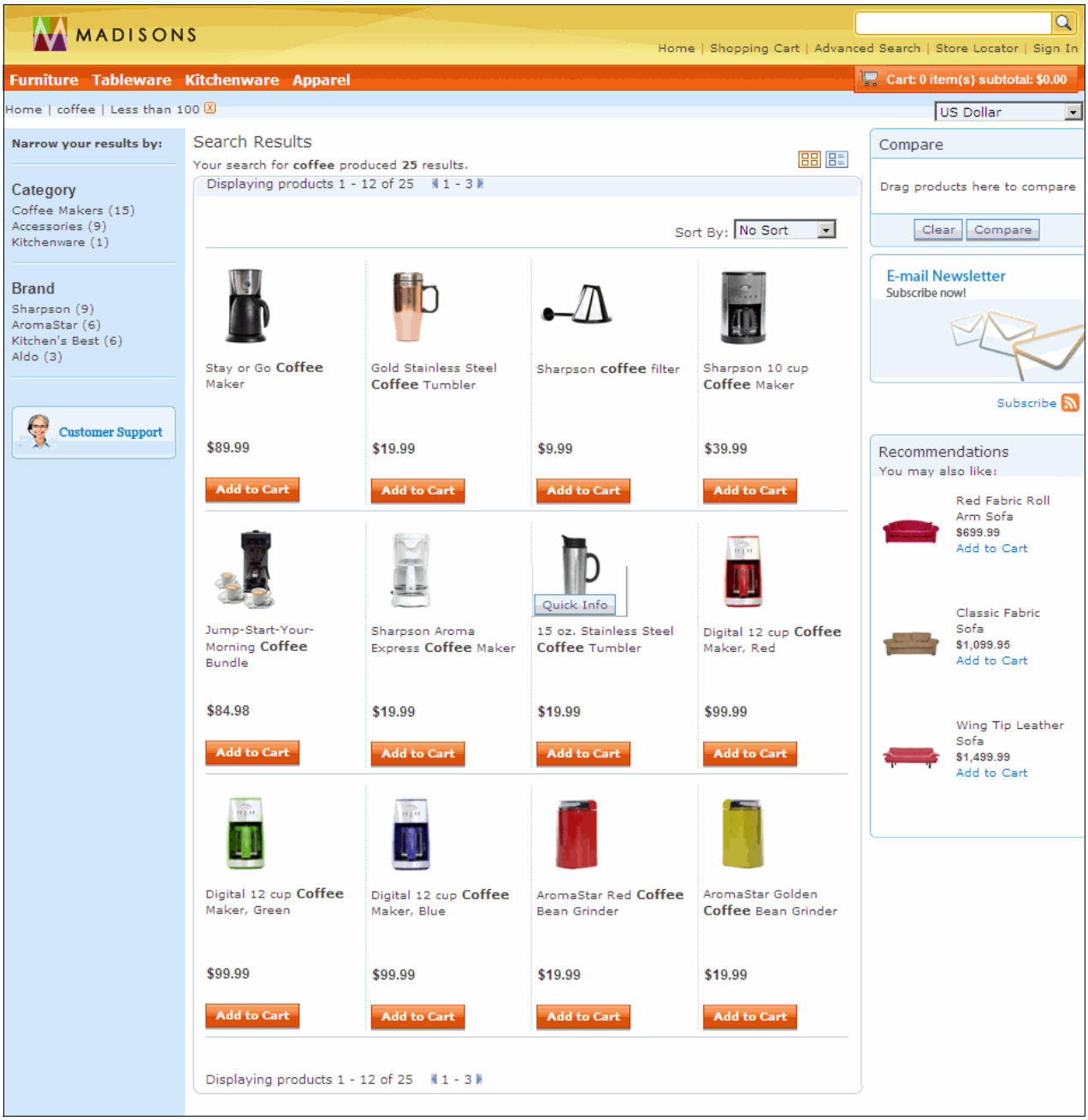


Figure 8-12 Search results with a price facet of Less than 100 selected

Selecting the facet **Kitchen's Best** further filters the data by brand and displays six products, as shown in Figure 8-13.

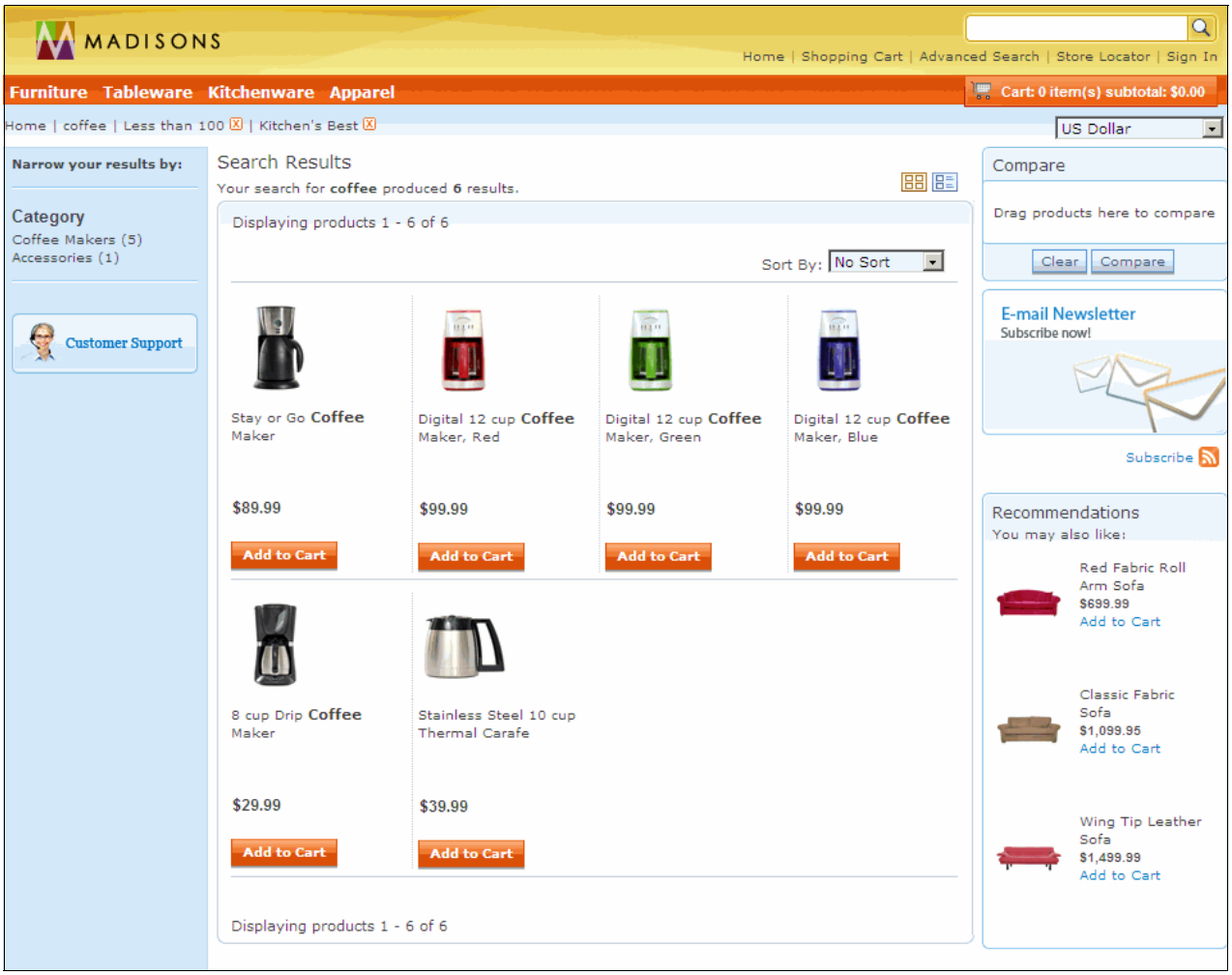


Figure 8-13 Search results with brand facet Kitchen's Best selected

You can configure the facet display in `wc-server.xml`. The `<_config:facets>` node declares a list of search index fields to be used for faceting and their associated faceting behavior at run time. The following parameters are available to customize the facet display:

- `sort`: This parameter determines the ordering of the facet field constraints. Assigning a value to `count` means to sort the constraints by the count (highest count first) and `index` means to return the constraints sorted in their index order. The default value is `count`.
- `mincount`: This parameter indicates the minimum counts for facet fields that need to be included in the response. The default value is 0.
- `limit`: This parameter indicates the maximum number of constraint counts that need to be returned for the facet fields. A negative value means unlimited. The default value is 100.

Example 8-2 shows the default values that are assigned for these variables.

Example 8-2 Facet configuration

```
<_config:facets>
<_config:param name="sort" value="count" />
<_config:param name="minCount" value="1" />
```

```
<_config:param name="limit" value="10" />
- <_config:category scope="all">
<_config:facet
converter="com.ibm.commerce.catalog.facade.server.services.search.metadata.solr.So
lrSearchCategoryFacetMetaDataConverter" name="parentCatgroup_id_facet" />
<_config:facet name="*" />
  </_config:category>
  </_config:facets>
```

8.8 Interactive breadcrumb trail

The breadcrumb trail maintains the current navigation structure as the shopper navigates throughout the store. It contains the current navigation facets that are selected by the shopper, with the category always shown first.

Facets previously selected by the shopper are shown in the breadcrumb trail. Seeing the previously selected facets enables the shopper to quickly identify what facets have been selected up to this point. These facets can also be cancelled by clicking [X] (remove) to the right of the facet in the breadcrumb trail, which will return more general search results.

For example, when the shopper searches for `coffee`, the system returns 31 products. The shopper then selects the facet price **Less than 100** and then selects the facet brand name **Kitchen's Best** from the search results.

The Search Results page contains the breadcrumbs, as shown in Figure 8-14.

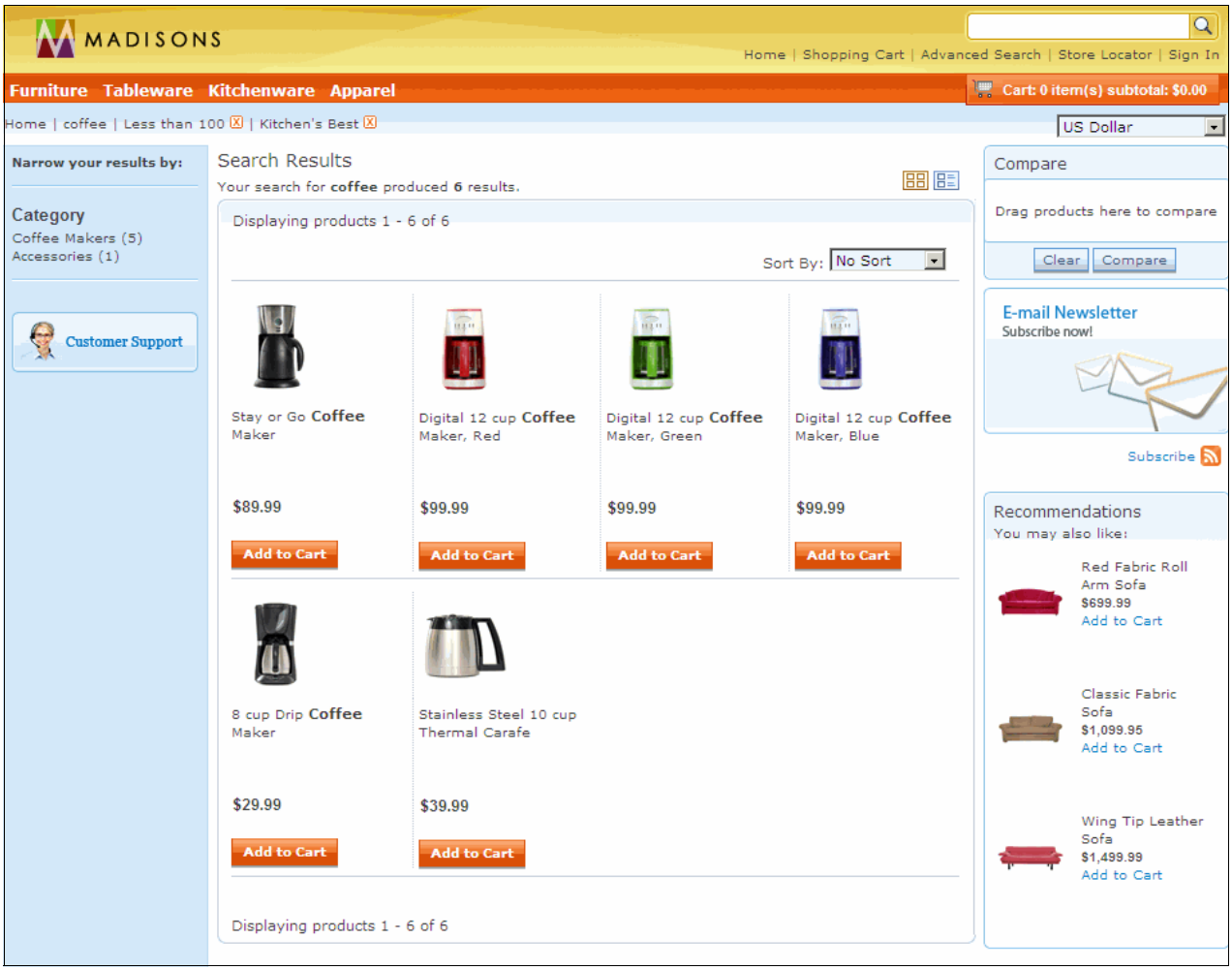


Figure 8-14 Selected facets are shown in the breadcrumb trail in the upper-left corner of the Search Results page

The shopper can now remove the facet `Less than 100` by clicking [**X**] in the breadcrumb trail, as shown in Figure 8-15.

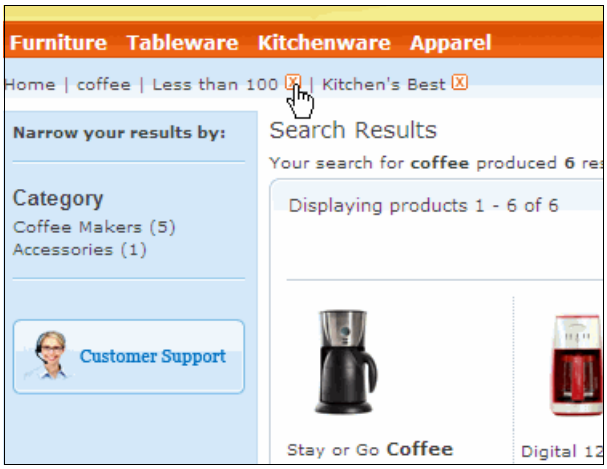


Figure 8-15 Removing the Less than 100 facet

8.9 Search term highlight

The search term highlight enables customers to browse easily where their submitted search term appears for each search result. Figure 8-16 shows the highlighting of the search term `sofa`.

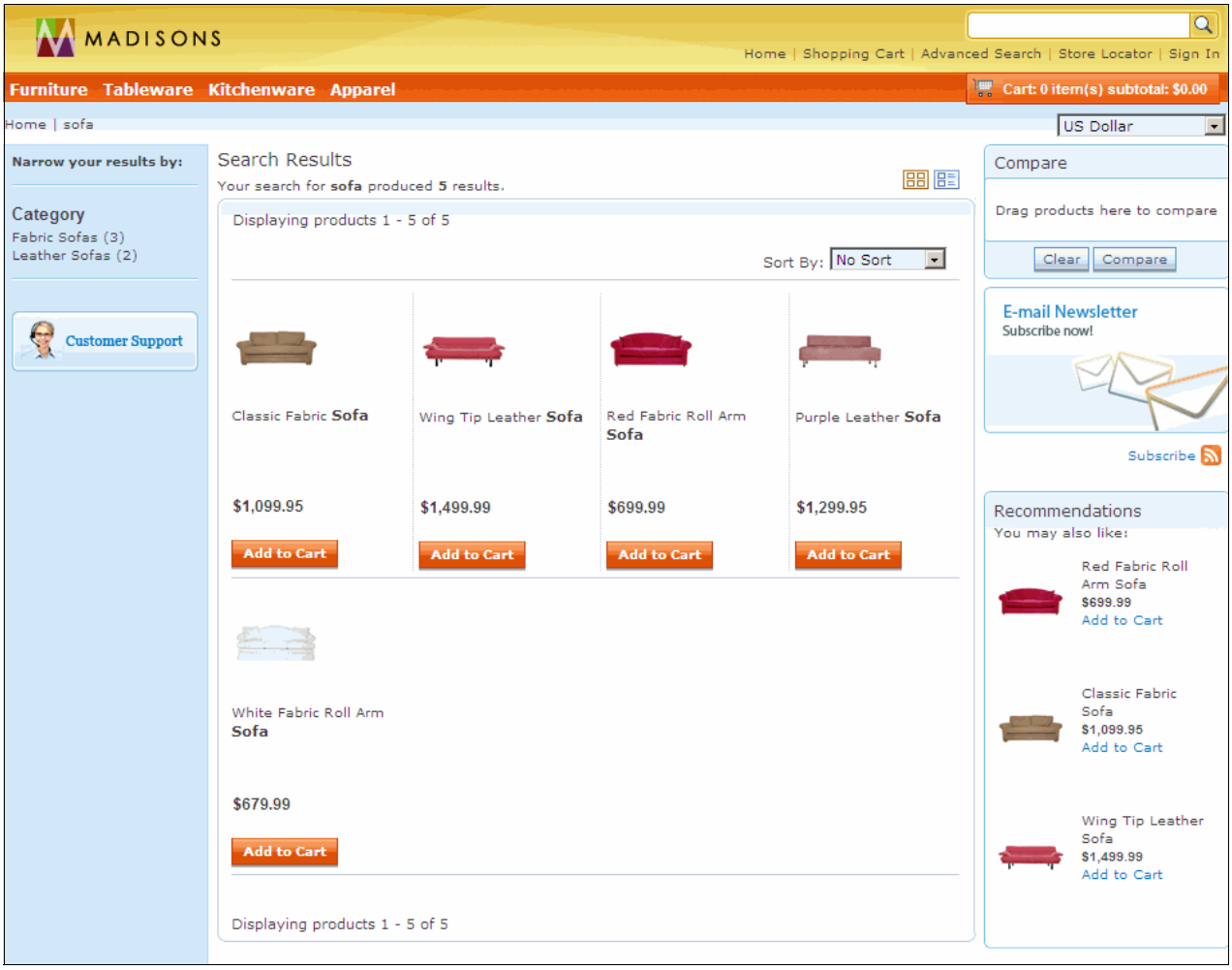


Figure 8-16 Search term sofa highlighted in search results

You configure the list of search index fields to use for highlighting and their associated highlighting behavior at run time in `wc-search.xml`, as shown in Example 8-3.

Example 8-3 Search term highlight configuration

```
<_config:highlight simplePost="&lt;/span>&lt;/strong>"
simplePre="&lt;strong>&lt;span class=font2>"/>
```

8.10 Multiple views

The Search Results page has two display modes:

- Grid view
- Details view

8.10.1 Grid view

The grid view allows more results to be displayed on a page, reducing the total number of results pages. It includes high-level information about the product, such as the name and price, as shown in Figure 8-17. Hovering the mouse over the result allows a quick view of the product.

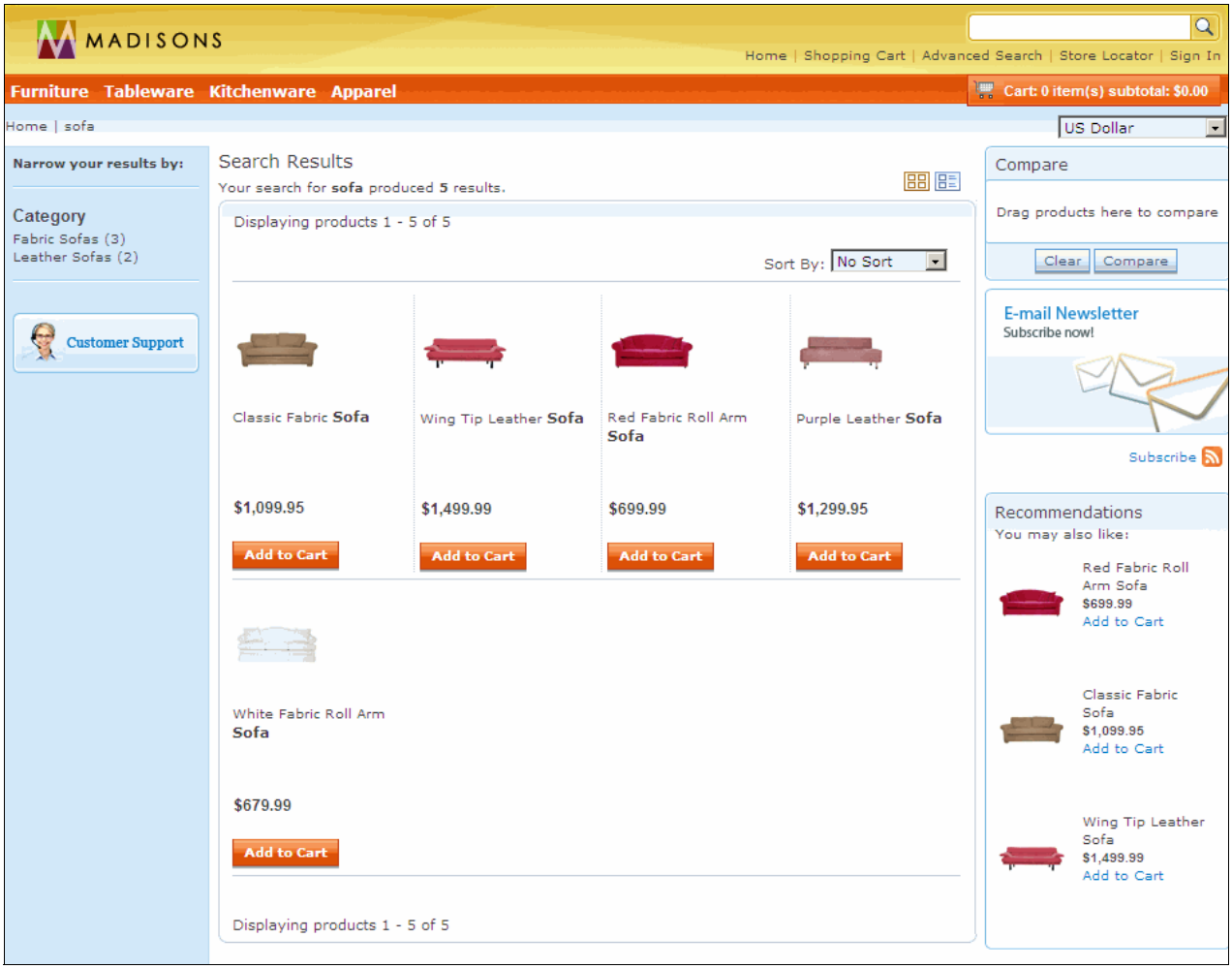


Figure 8-17 Search Results grid view

To change the view, click the details view image, as shown in Figure 8-18.

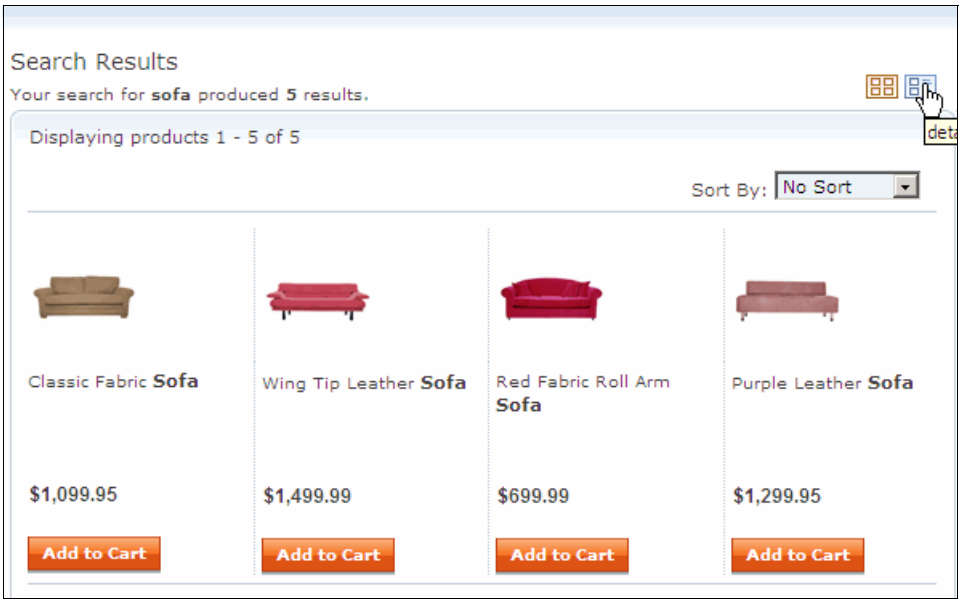


Figure 8-18 Switching between views

8.10.2 Details view

The details view shows more information for the search result products, such as name, short description, and price. It also shows the search term highlighting and unstructured content results, as shown in Figure 8-19 on page 224.

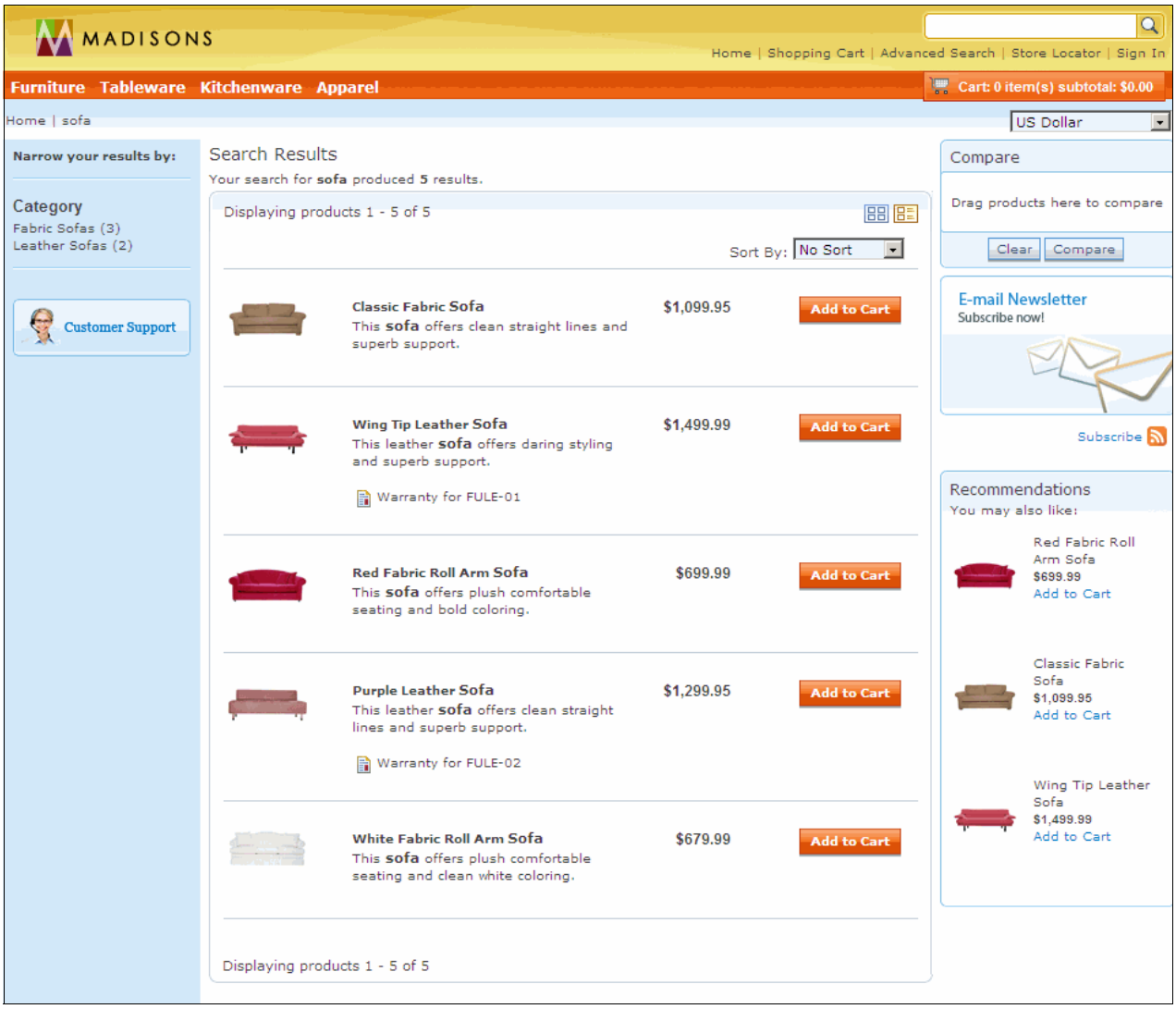


Figure 8-19 Search results details view

8.11 Index or calculate displayed price

You define the display mode for showing prices in the storefront in `wc-search.xml`. The prices can be computed, indexed, or a mixture of both:

- Computed: Prices only for the current page are calculated at run time and the page hides the price facets. The system shows the computed price in each result. It hides the price range search in the Advanced Search page. *Use the computed price when prices are not populated in the search index*. Assign a value of 0 in the configuration file `wc-search.xml`.
- Indexed: In this mode, all prices are retrieved from the search index and the system shows the price facets with the price range configured in the facet configuration table. It shows the indexed price in each search result. It shows the price range search in the Advanced Search page. *Use this mode when the prices are populated in the search index*. Assign a value to 1 in the configuration file `wc-search.xml`.
- Mixed: In this mode, the search result contains the computed prices and the price facets use the indexed prices. The system shows the price facets with the price range configured

in the facet configuration table. It shows the computed price in each result. It shows the price range search in the Advanced Search page. *Use this mode when the prices for all supported currencies are populated in the search index*. Assign a value of 2 in the configuration file `wc-search.xml`.

Example 8-4 shows configuring the prices to be indexed.

Example 8-4 Prices are indexed

```
<_config:param name="price" value="1"/>
```

8.12 Configurable sort options

You can define sorting for the search results in `wc-search.xml`. The sort section is for defining the sorting options and their corresponding values, which can be used directly from the storefront (Example 8-5).

Example 8-5 Sort options

```
<_config:sort inherits="false">
            <_config:field name="1" value="mfName_ntk_cs asc"/>
            <_config:field name="2" value="name_ntk asc"/>
            <_config:field name="3" value="price_* asc"/>
            <_config:field name="4" value="price_* desc"/>
</_config:sort>
```

8.13 Drag and drop results

Shoppers can drag search results to the Compare view or to the shopping cart.

For example, shoppers can search for `sofa`. From the Search Results page, drag and drop **Classic Fabric Sofa** to the Compare view, as shown in Figure 8-20.

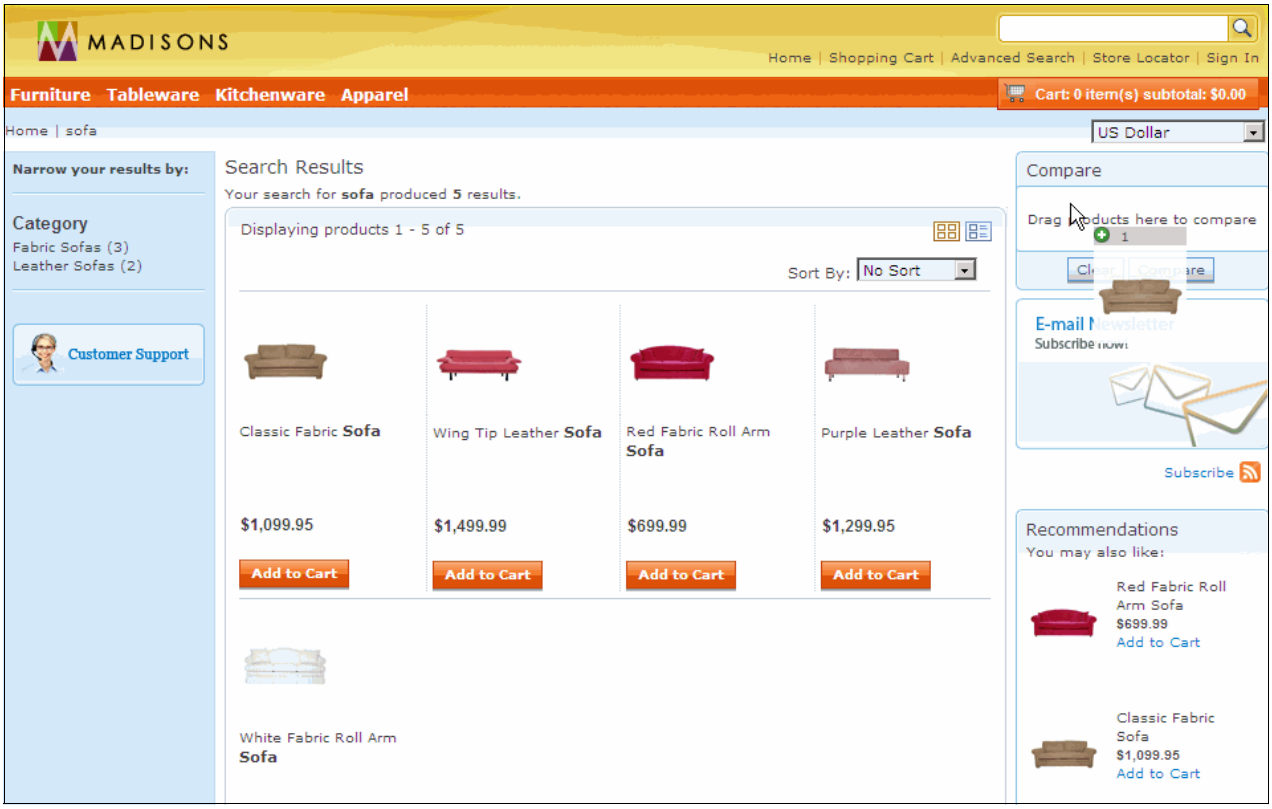


Figure 8-20 Drag and drop Classic Fabric Sofa to the Compare view

Now, from the Search Results page, drag and drop **Wing Tip Leather Sofa** to the quick cart, as shown in Figure 8-21. This action adds the product to the shopping cart.

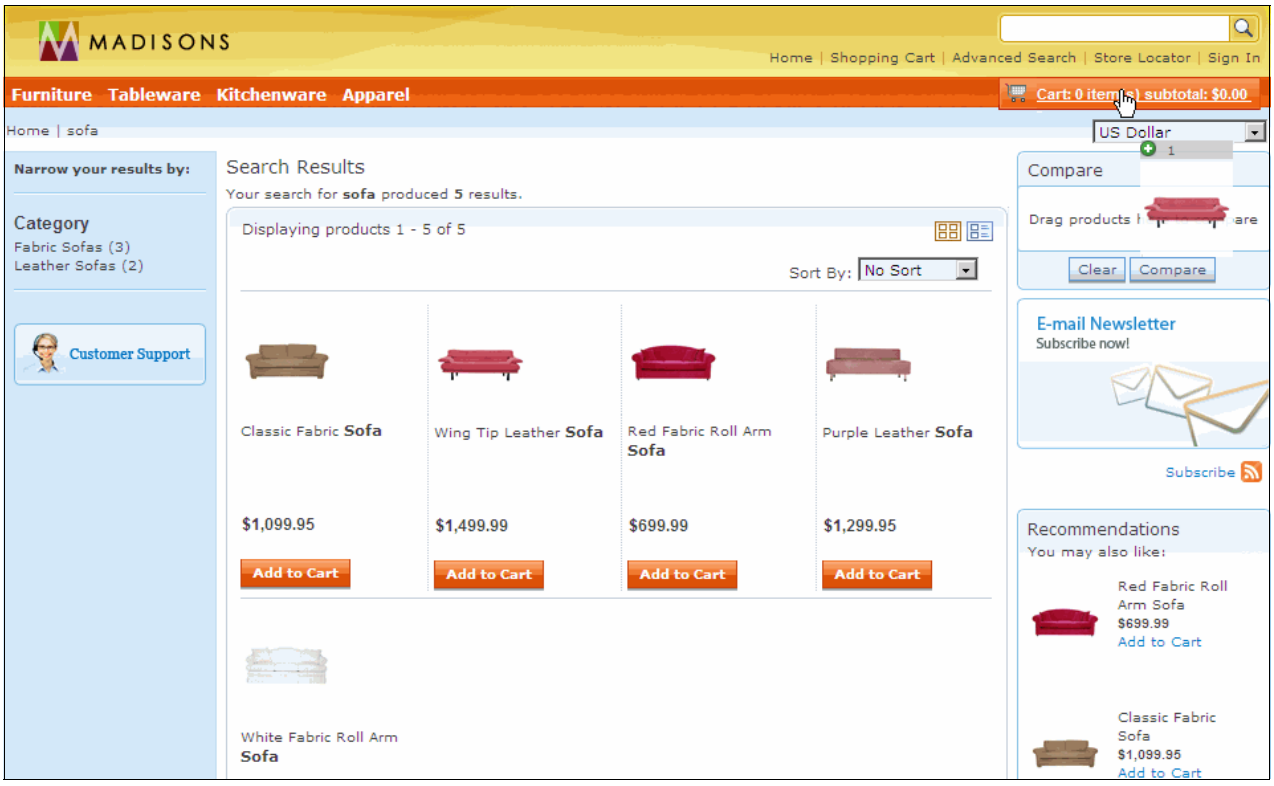


Figure 8-21 Drag and drop Wing Tip Leather Sofa to the quick cart

8.14 Bookmark results

You can bookmark the search results pages like any other web pages. Shoppers can view these bookmarked pages directly by selecting the bookmark. For storefronts that require the shopper to log in before viewing the catalog, the shopper can be prompted to log in before seeing the search results bookmark.

Part 3

Business-to-business (B2B) scenario

In this part, we discuss the entitlement feature that is provided through the WebSphere Commerce search framework. We also discuss a B2B scenario and provide the implementation using catalog filters and contracts.

A *catalog filter* is a set of user-defined conditions that calculate the categories and catalog entries to which a customer is entitled. The catalog filter provides an entitlement search expression as part of the search criteria. This entitlement check uses the WebSphere Commerce search framework. It is integrated with the search solution to provide fast and dynamic catalog filtering. Without customization, the search framework uses Solr.

You need to perform the following prerequisites to use catalog filters:

1. Install WebSphere Commerce V7.0 Fix Pack 2 and WebSphere Commerce V7.0 Feature Pack 2.
2. Enable the Foundation and Management Center feature.
3. Publish the stores and corresponding enhancement SAR (store archive) files.
4. Configure search.
5. Enable search:
   a. Enable product entitlement:
      i. Enable the storefront for search.
      ii. Enable the search-based navigation flow.
   b. Create the product-sensitive policies, such as “Use in price rules”:
      i. Enable the storefront for search.

By default, all stores have store default filters created and assigned.

Building search expressions

Figure 8-22 shows the interface that is used to build up the search expression in String format using contract IDs (either from input, or from the business context if there is no input contract).

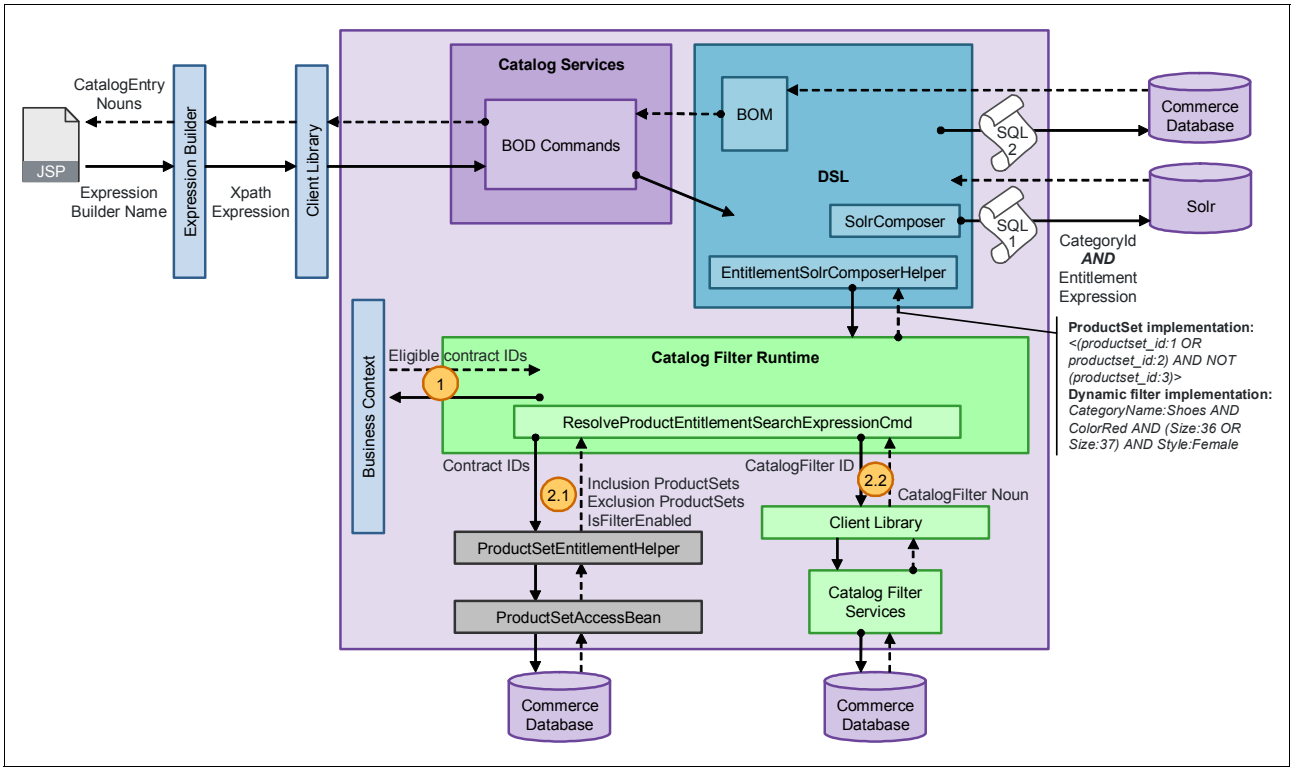


Figure 8-22 Catalog filter overview

The internal logic determines whether ProductSet TC or CatalogFilter TC is used, and then, it composes the search expression based on it.

Business users can create catalog filters that are based on categories, catalog entries, product properties, and attribute dictionary attribute conditions. E-site managers can assign catalog filters to e-site stores. Business-to-business (B2B) sales managers can assign catalog filters to default or buyer contracts.

When creating catalog filters, you can combine one or more filtering criteria to achieve your business goals. You can include or exclude categories from your catalog. If you include a category, any catalog entry and child category under this category are included, unless you explicitly exclude the catalog entry or category. You can apply conditions on the categories that are added to your catalog filter. The condition can be composed of attribute dictionary attributes or catalog entry properties. You can include or exclude catalog entries from your catalog. The catalog entry-based criteria takes higher precedence over the other filtering criteria. You can allow separate customers to view and purchase separate catalog entries on separate stores by assigning catalog filters to contracts. When you assign catalog filters to a contract, the customers that are shopping under the contract can browse, search for, and purchase the catalog entries that are defined in the catalog filter. For example, in a B2B business model, you can create catalog filters and then assign separate catalog filters to each business contract. The customers shopping under the specific contract only can see the catalog entries to which they are entitled.

Product entitlement

Product entitlement consists of the following functions:

- Assign catalog filters to contracts in Accelerator.
- Product entitlement within contracts:
  - Base contract
  - B2B buyer contract
  - Store default contract
- For B2B direct stores, you can assign a catalog filter to the following contracts:
  - The default contract for the store
  - A base contract
  - A customer contract
- For e-site stores (B2B direct and consumer direct), you can also assign a catalog filter to these contracts:
  - The default base contract of the storefront asset store
  - The default contracts of extended site stores

  Important: Catalog filters cannot be assigned to hosting contracts. The channel manager role, which is responsible for hosting contracts, does not have access to the catalog filter and pricing tool.

- For business-to-consumer (B2C) direct stores, assign the catalog filter to the default store contract.

In this part of the book, we describe this scenario. EliteFEP2 is a B2B automotive parts store that sells products, such as brake components, electrical systems, suspension components, transmission components, and accessories. The EliteFEP2 storefront has to provide separate contract entitlements based on the organization to which the user belongs:

- Customers under the ITSO Distributor A company must be able to view and order all products, except for products that are made by the Brakin Brothers company.
- Customers under the ITSO Generic distributor organization must not be able to view and order products under Transmission components, except for one product, which is Spur gears (part number A00001560).
- Customers under the ITSO Electrical Systems distributor organization must be able to view and order all products that are under Electrical systems only (exclude all other categories).

9

Catalog entitlement

This chapter provides a step-by-step implementation of the scenario that is introduced in Part 3, “Business-to-business (B2B) scenario” on page 229.

This chapter assumes that the users, organizations, and contracts, which are shown in Table 9-1, Table 9-2, and Table 9-3, are already created in the system.

Table 9-1 Organizations

Organization name	Parent organization	Organization type
ITSO Generic	EliteFEP2 Organization	Organization
ITSO Distributor A	EliteFEP2 Organization	Organization
ITSO Electrical Systems	EliteFEP2 Organization	Organization

Table 9-2 Users

User ID	Organization	Role
Jim	ITSO Generic	Registered Customer
Chris	ITSO Distributor A	Registered Customer
Nikit	ITSO Electrical Systems	Registered Customer

All contracts are created under the *Buyer A Organization Account for EliteFEP2* account.

Table 9-3 Contracts

Contract name	Participants (Organizations)
ITSO Generic contract	ITSO Generic
ITSO Distributor A contract	ITSO Distributor A
ITSO Electrical Systems distributor contract	ITSO Electrical Systems

9.1 Catalog entitlement

Before assigning catalog filters to the contracts, we need to create three catalog filters to satisfy the requirement. Perform these steps to create the catalog filters:

1. Log on to Management Center:

   http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.management-center.doc/tasks/ttflogon.htm

2. Open the **Catalog Filter and Pricing** tool, as shown in Figure 9-1.

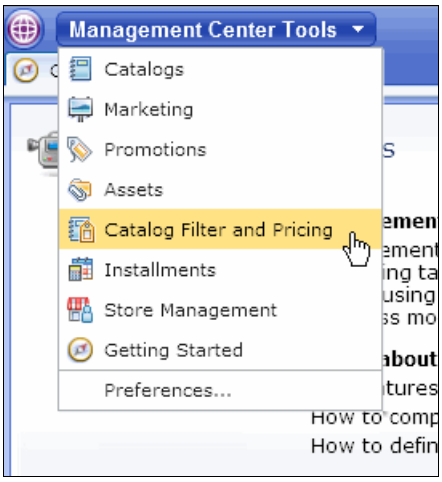


Figure 9-1 Open Catalog Filter and Pricing tool

3. Create a catalog filter 1 (`CF1`) to include all products in the master catalog, except for products that are made by `Brakin Brothers`:
   a. Create a new catalog filter by clicking the **New** icon and selecting **Catalog Filter**.
   b. Specify the following fields under General Properties, as shown in Figure 9-2:
      i. For Name, type `CF1`.
      ii. For Description, type `Include all products except those with manufacturer name 'Brakin Brothers'`.

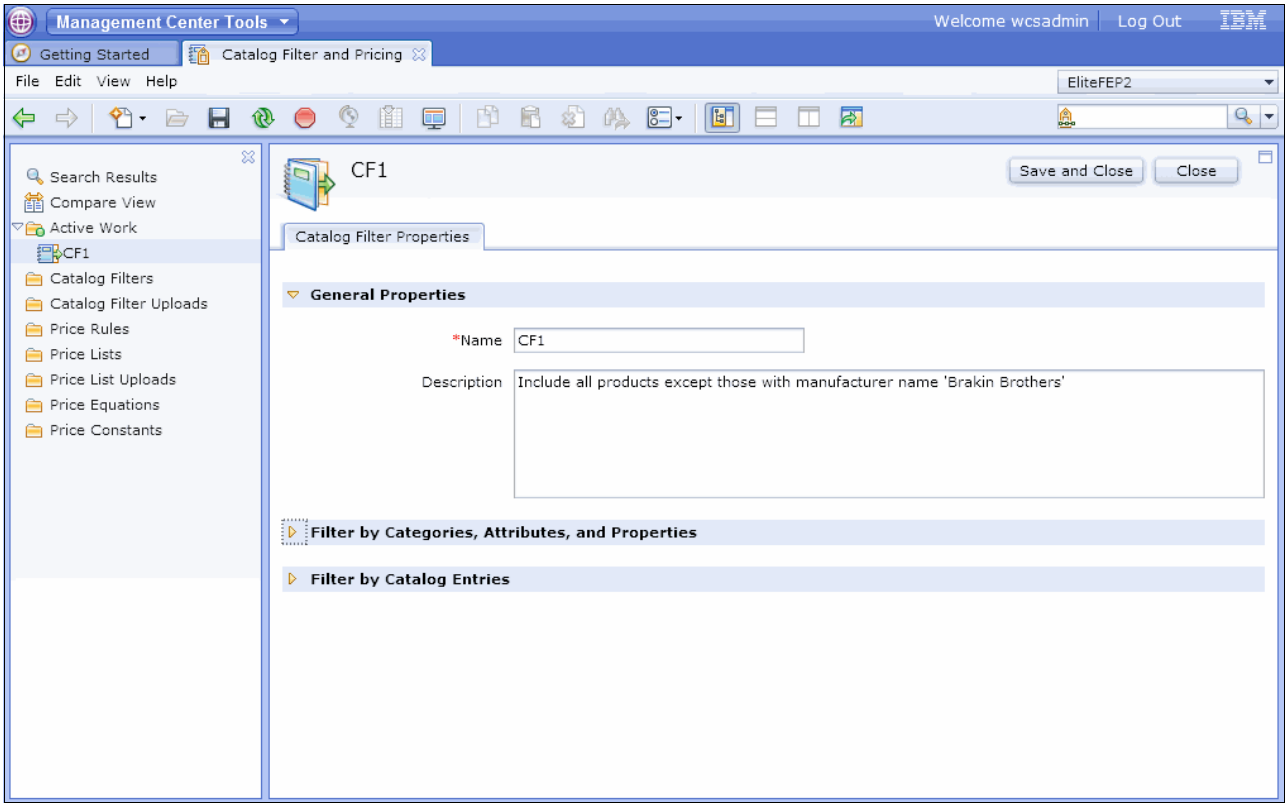


Figure 9-2 Creating catalog filter 1 (CF1)

   c. Expand **Filter by Categories, Attributes, and Properties**.
   d. Select **Tree view** mode.

e. Right-click **EliteFEP2** in Categories and select **Include**, as shown in Figure 9-3.

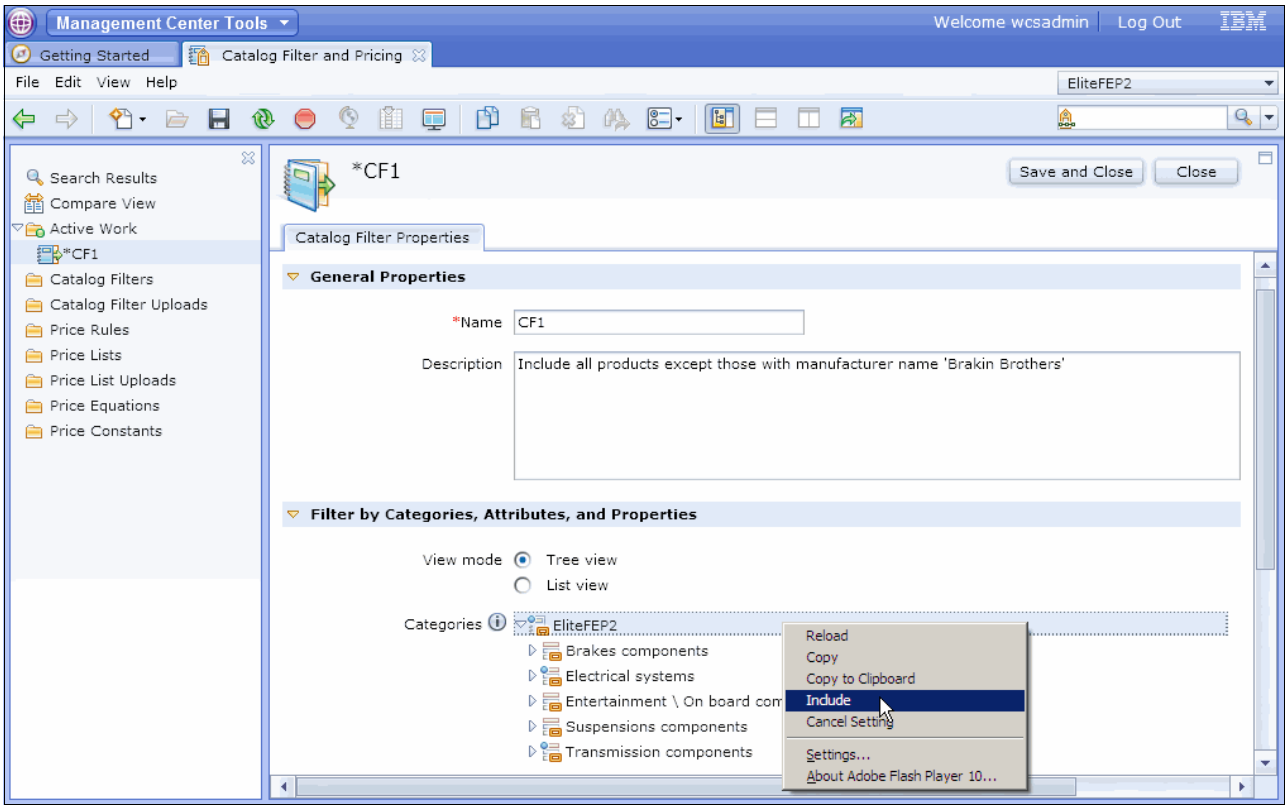


Figure 9-3 Include the master catalog

f. In the Condition groups, create a new condition group called `CG1`.

g. Click the condition group **CG1** and add the following attribute condition:

   i. Click a new property in Attributes and properties conditions, as shown in Figure 9-4.

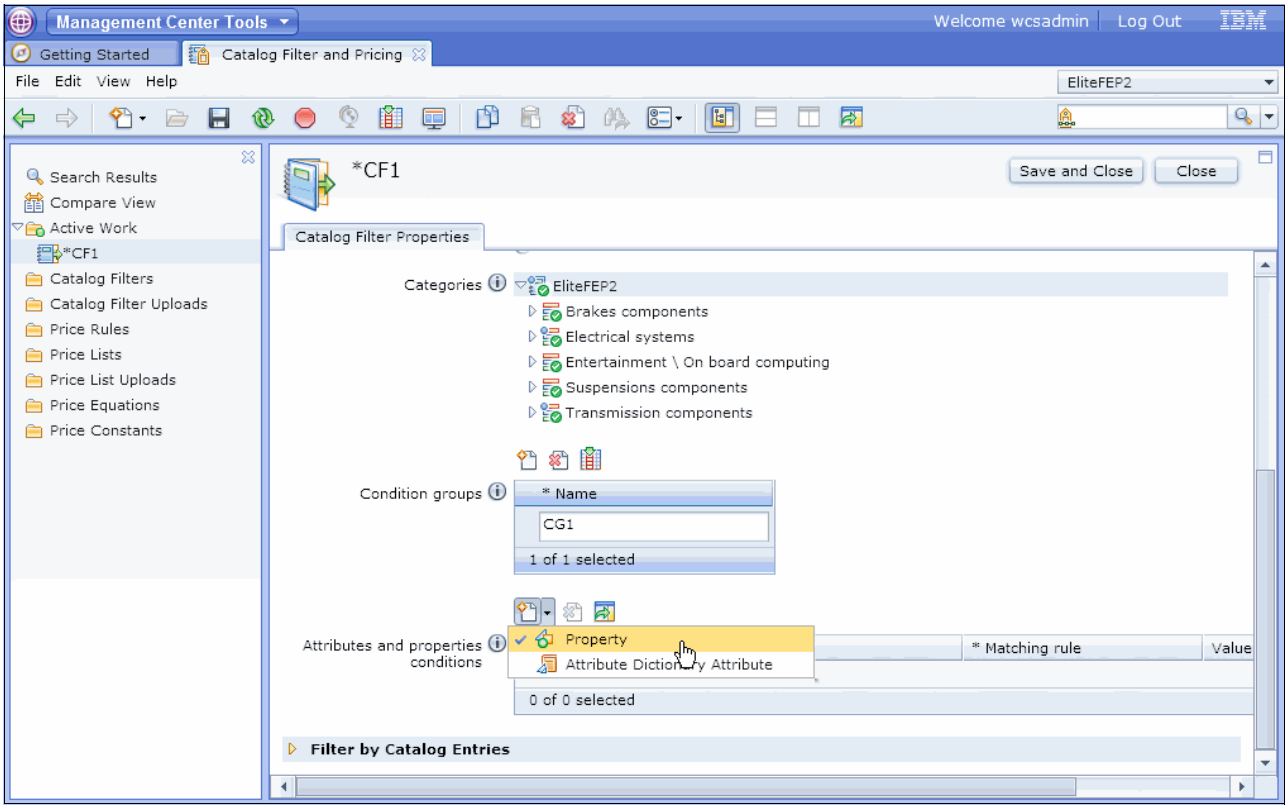


Figure 9-4 Creating a new attribute condition

ii. Select **Manufacturer Name** under the Name column.

iii. Select Matching rule as **not equals**.

iv. For Value, enter `Brakin Brothers`, as shown in Figure 9-5.

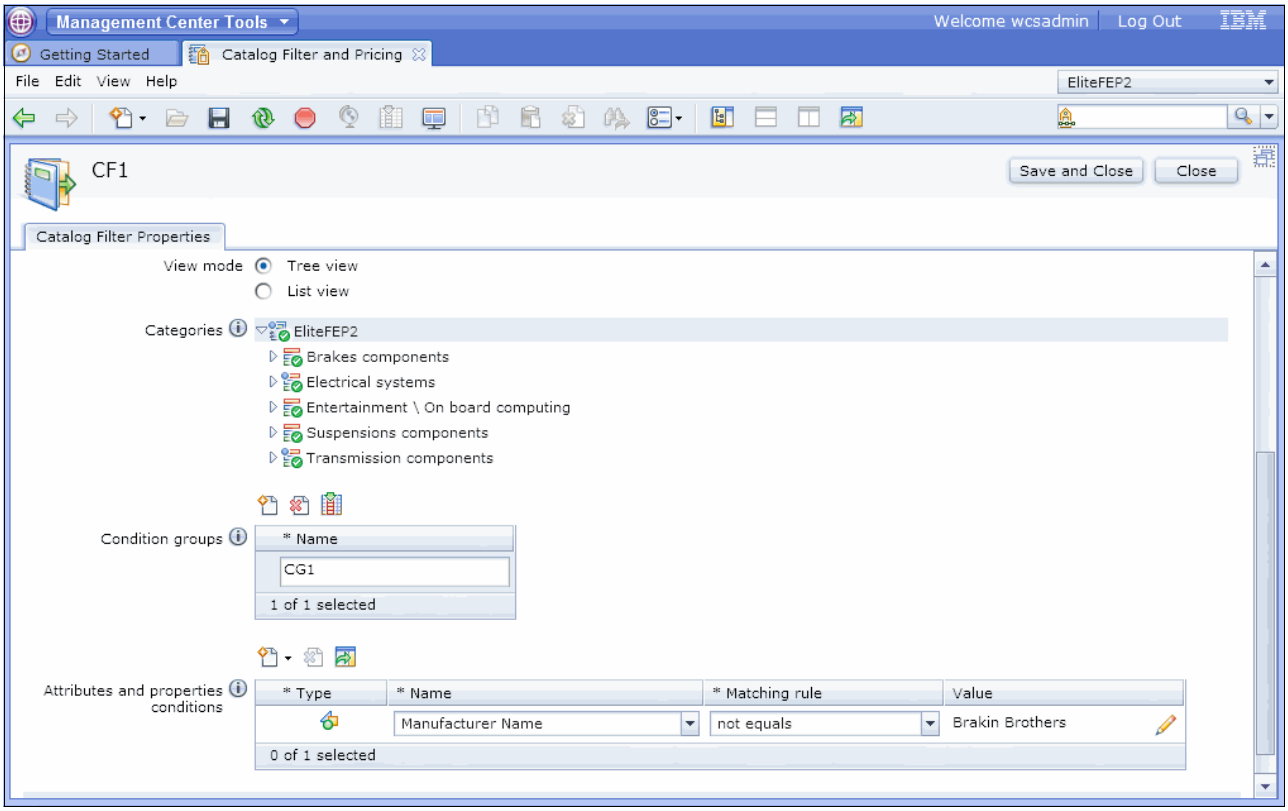


Figure 9-5 Selecting the property conditions

h. Click **Save and Close**.

4. Create catalog filter 2 (`CF2`) to exclude all products under the category Transmission components, except for one product called Spur gears (part number A00001560):

   a. Create a new catalog filter by clicking the **New** icon and selecting **Catalog Filter**.

   b. Specify the following fields under General Properties:

      i. For Name, enter `CF2`.

      ii. For Description, enter `Exclude all products under Transmission components except Spur gears`.

   c. Expand **Filter by Categories, Attributes, and Properties**.

   d. Select **Tree view** mode.

   e. Right-click **EliteFEP2** in Categories and select **Include**.

f. Right-click the **Transmission components** category and select **Exclude**, as shown in Figure 9-6.

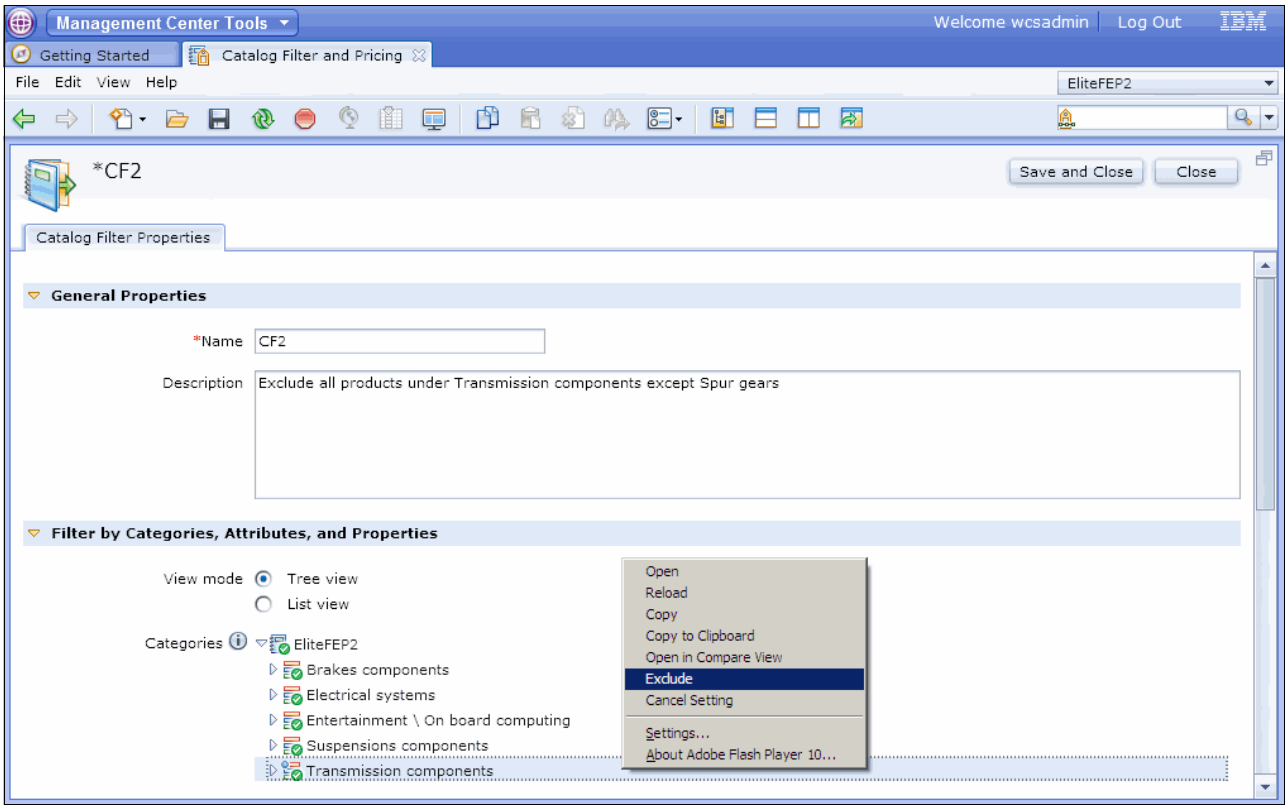


Figure 9-6 Excluding the Transmission components category

g. Expand **Filter by Catalog Entries**.

h. In the Catalog entries to include section, enter `A00001560` for Code and click **Find and Add**, as shown in Figure 9-7.

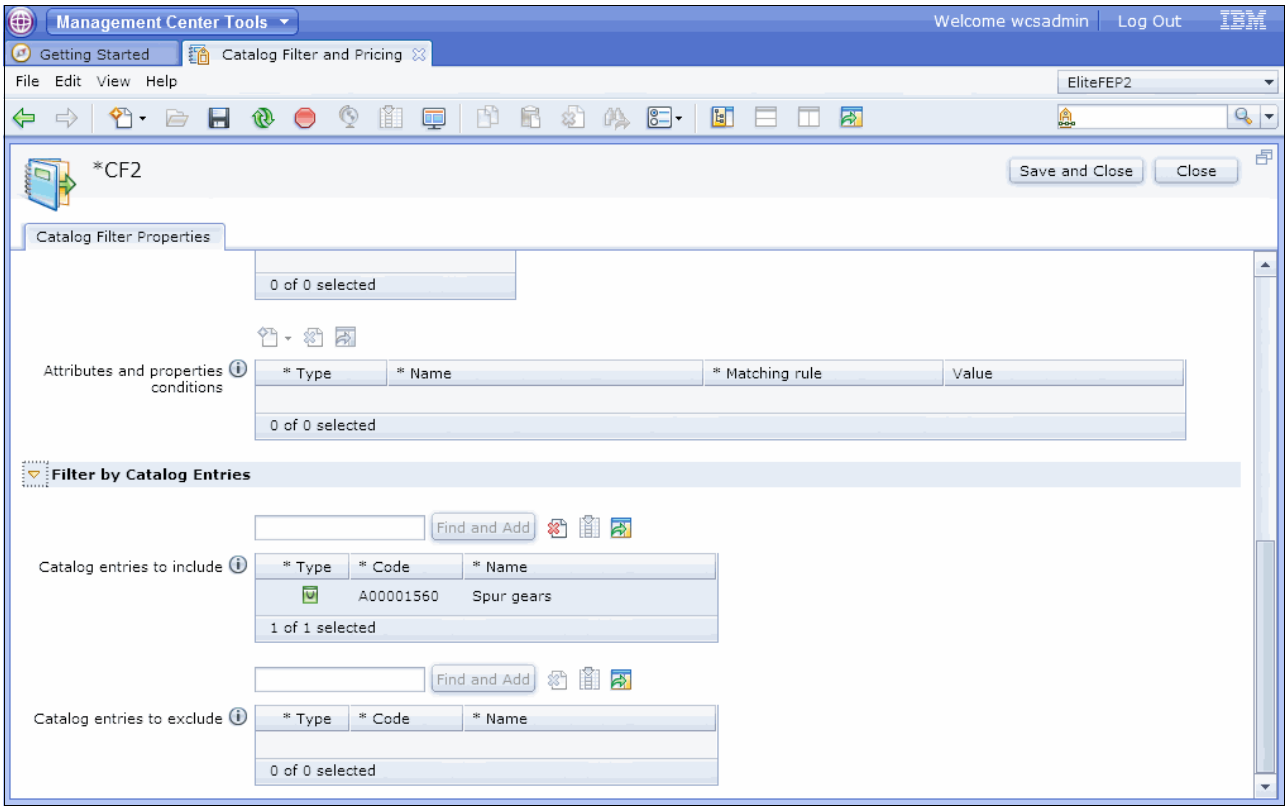


Figure 9-7 Adding the catalog entry to include

i. Click **Save and Close**.

5. Create catalog filter 3 (`CF3`) to include all products under the Electrical systems category and exclude all other categories:

   a. Create a new catalog filter by clicking the **New** icon and selecting **Catalog Filter**.

   b. Specify the following fields under General Properties:

      i. For Name, type `CF3`.

      ii. For Description, type `Include only Electrical systems` category products.

   c. Expand **Filter by Categories, Attributes, and Properties**.

   d. Select **Tree view** mode.

e. Right-click **EliteFEP2** in Categories and select **Exclude**, as shown in Figure 9-8.

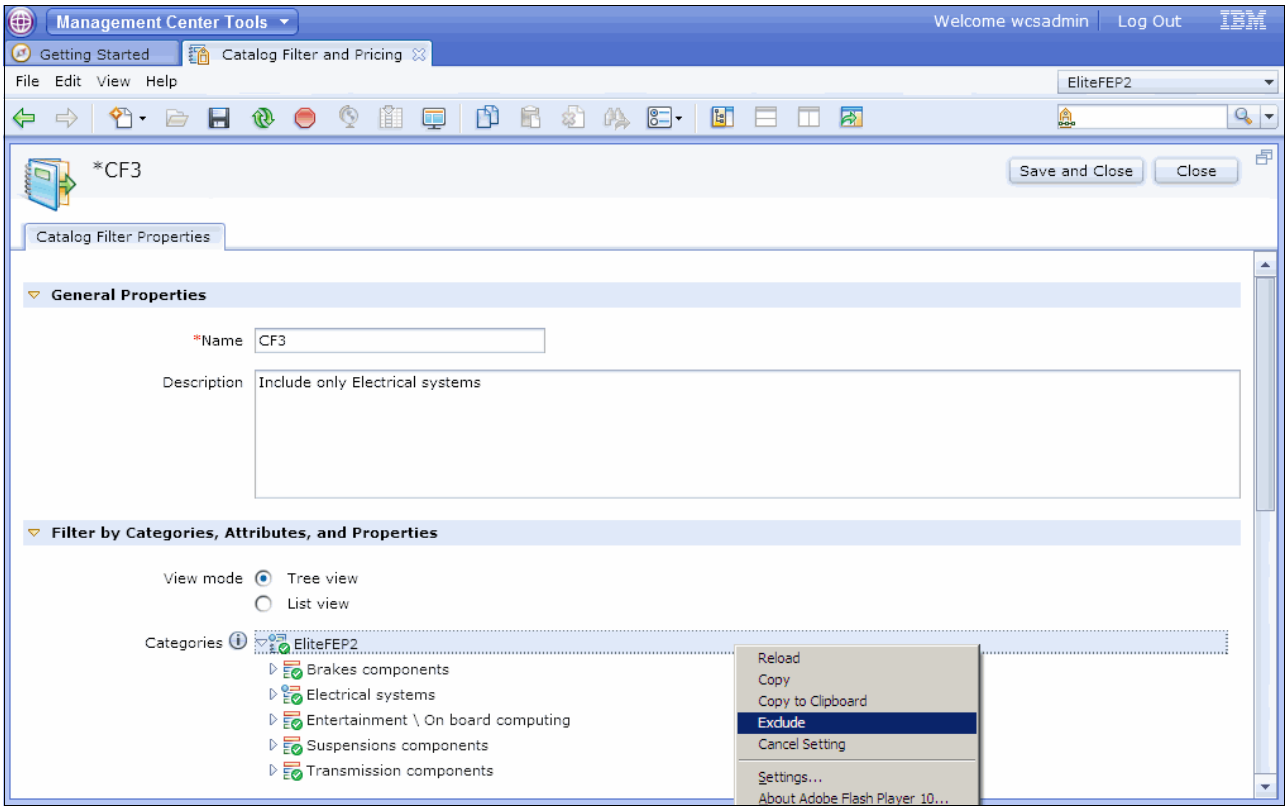


Figure 9-8 Excluding the master catalog

f. Right-click the **Electrical systems** category and select **Include**, as shown in Figure 9-9.

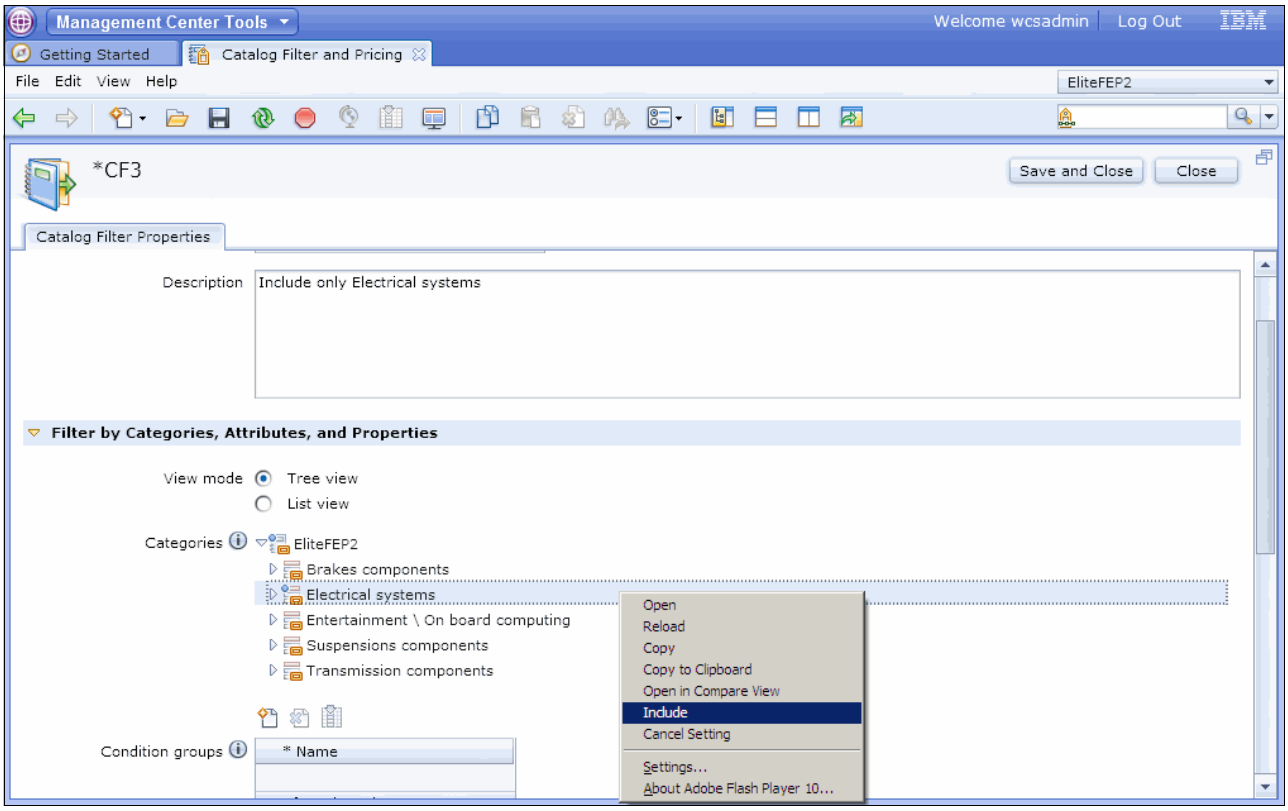


Figure 9-9 Including only the Electrical systems category

g. Click **Save and Close**.

To entitle certain customers to a subset of your catalog on the storefront, you must assign the catalog filter that we created to a contract by using WebSphere Commerce Accelerator. This requirement applies to all business models (business-to-business (B2B) direct stores, consumer direct stores, and extended site stores). As a result, customers shopping under the contract are only entitled to see and purchase the set of catalog entries that is defined in the catalog filter. A catalog filter is one of the terms and conditions that a contract can have.

Important: For a specific contract, only a single catalog filter can be in effect at one time. When assigning new catalog filter to a contract, the new catalog filter overrides the WebSphere Commerce Accelerator catalog filter.

Perform these steps to assign the catalog filters to contracts:

1. Log on to WebSphere Commerce Accelerator to assign these catalog filters to the contracts. For more information, see this website:

   `http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.user.doc/tasks/ttfopen.htm`

2. Select the **EliteFEP2** store from the stores list during the logon.

3. Select **Sales** → **Accounts**, as shown in Figure 9-10.

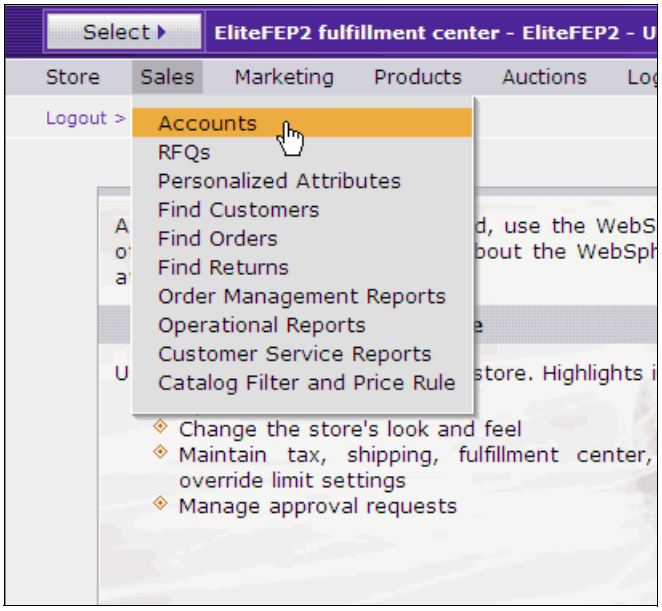


Figure 9-10 Opening the Contracts view

4. Select the business account, **Buyer A Organization**, that has the contract to which you want to assign the catalog filter, and then click **Contracts**, as shown in Figure 9-11.

Figure 9-11 Selecting the base contract

5. Select the target contract, **ITSO Distributor A contract**, and then click **Update Extended TC**, as shown in Figure 9-12.

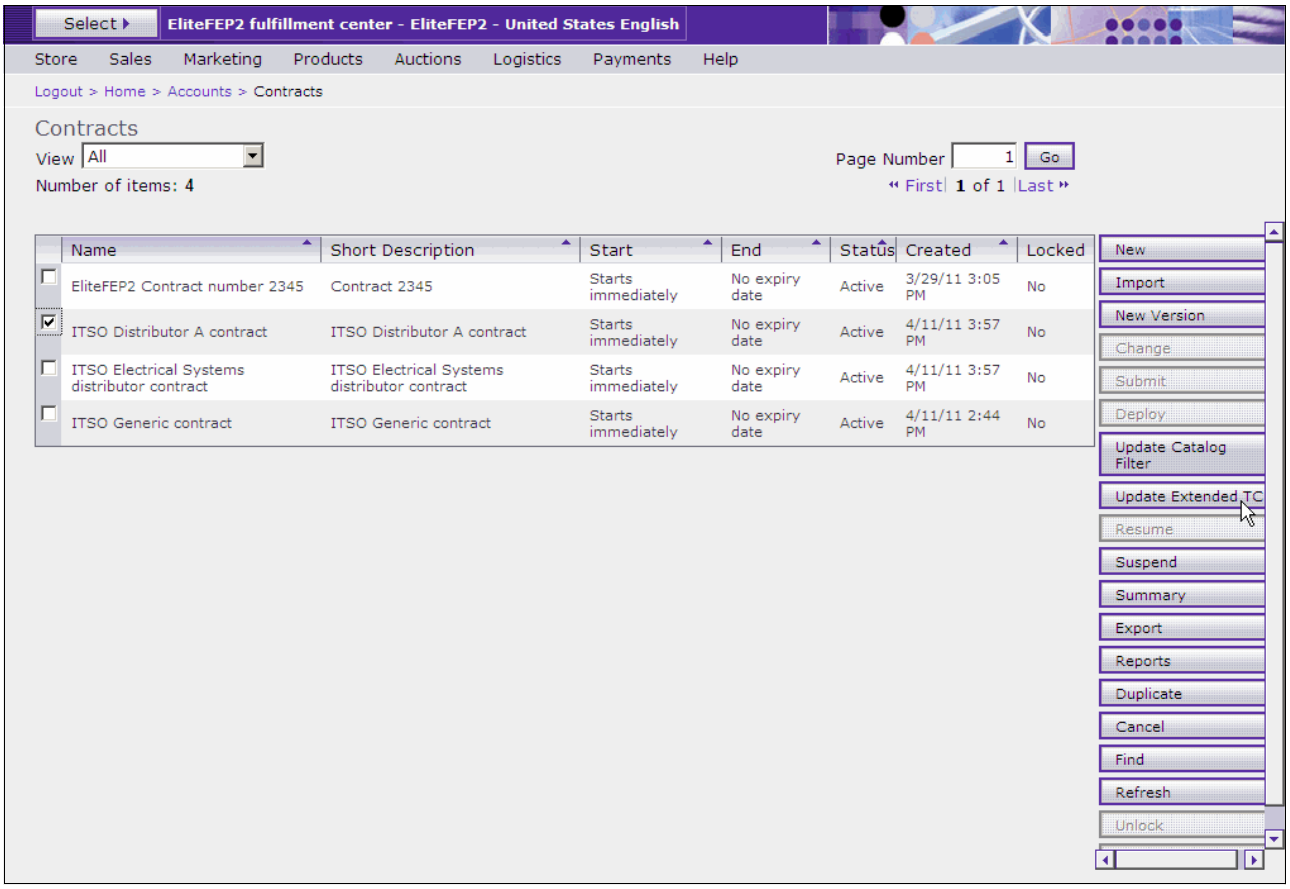


Figure 9-12 Updating contract terms and conditions

6. In the Catalog Filter tab, select **CF1** from the Property Value list box, and click **OK**, as shown in Figure 9-13.

Figure 9-13 Selecting the catalog filter to be assigned to the current contract

7. Select the target contract, **ITSO Generic contract**, and then click **Update Extended TC**.

8. Select **CF2** from the Property Value list box and click **OK**.
9. Select the target contract, **ITSO Electrical Systems contract**, and then click **Update Extended TC**.
10. Select **CF3** from the Property Value list box and click **OK**.

If Jim, who belongs to the ITSO Generic organization (ITSO Generic contract), logs into the storefront, Jim can see only one subcategory called gears under the Transmission category and it has only the product, Spur Gears, as shown in Figure 9-14.

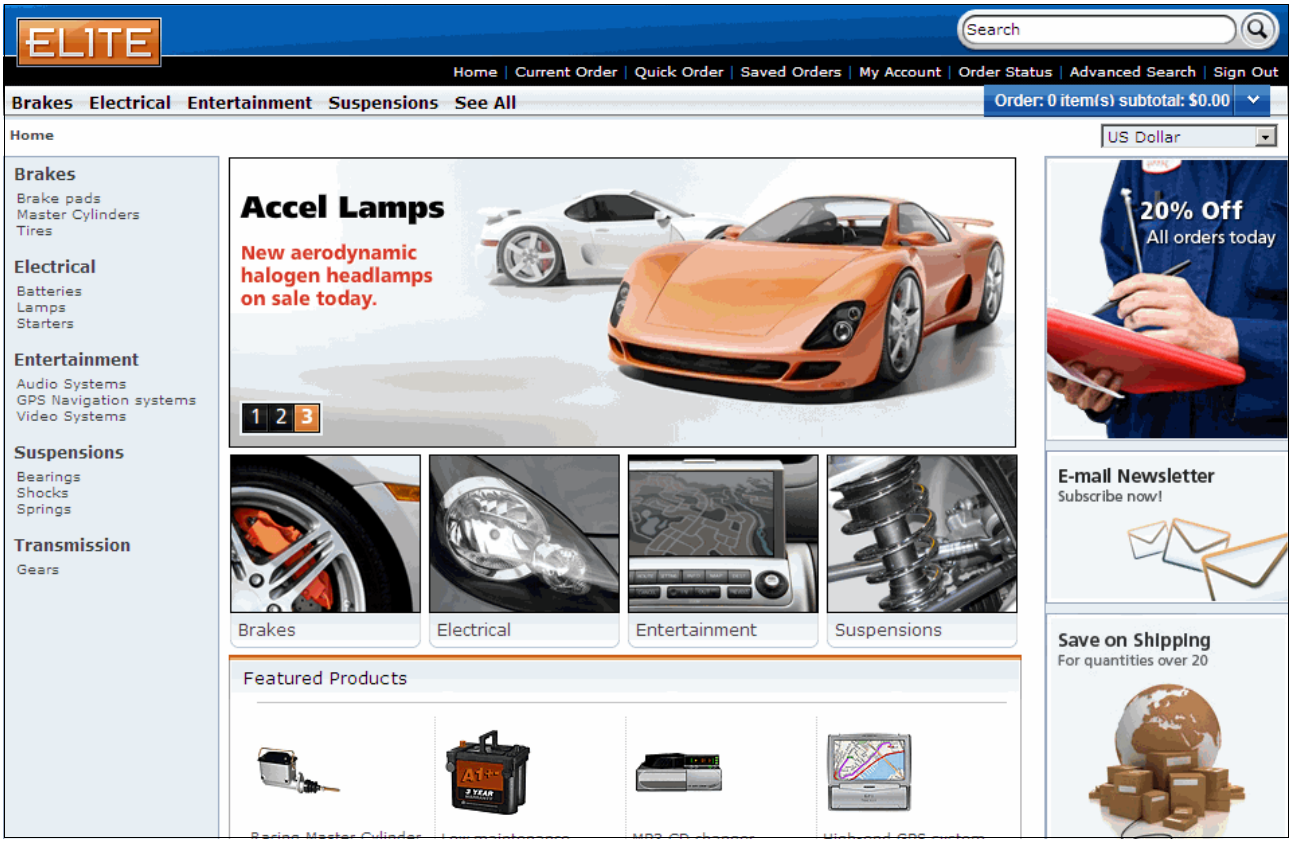


Figure 9-14 Storefront based on catalog filter CF1

If Chris, who belongs to the ITSO Distributor A organization (ITSO Distributor A contract), logs in to the storefront, Chris can see all the categories (except the products that are made by Brakin Brothers, Semi-metallic Brake pads and Organic Brake pads) under the Brake Pads subcategory, as shown in Figure 9-15.

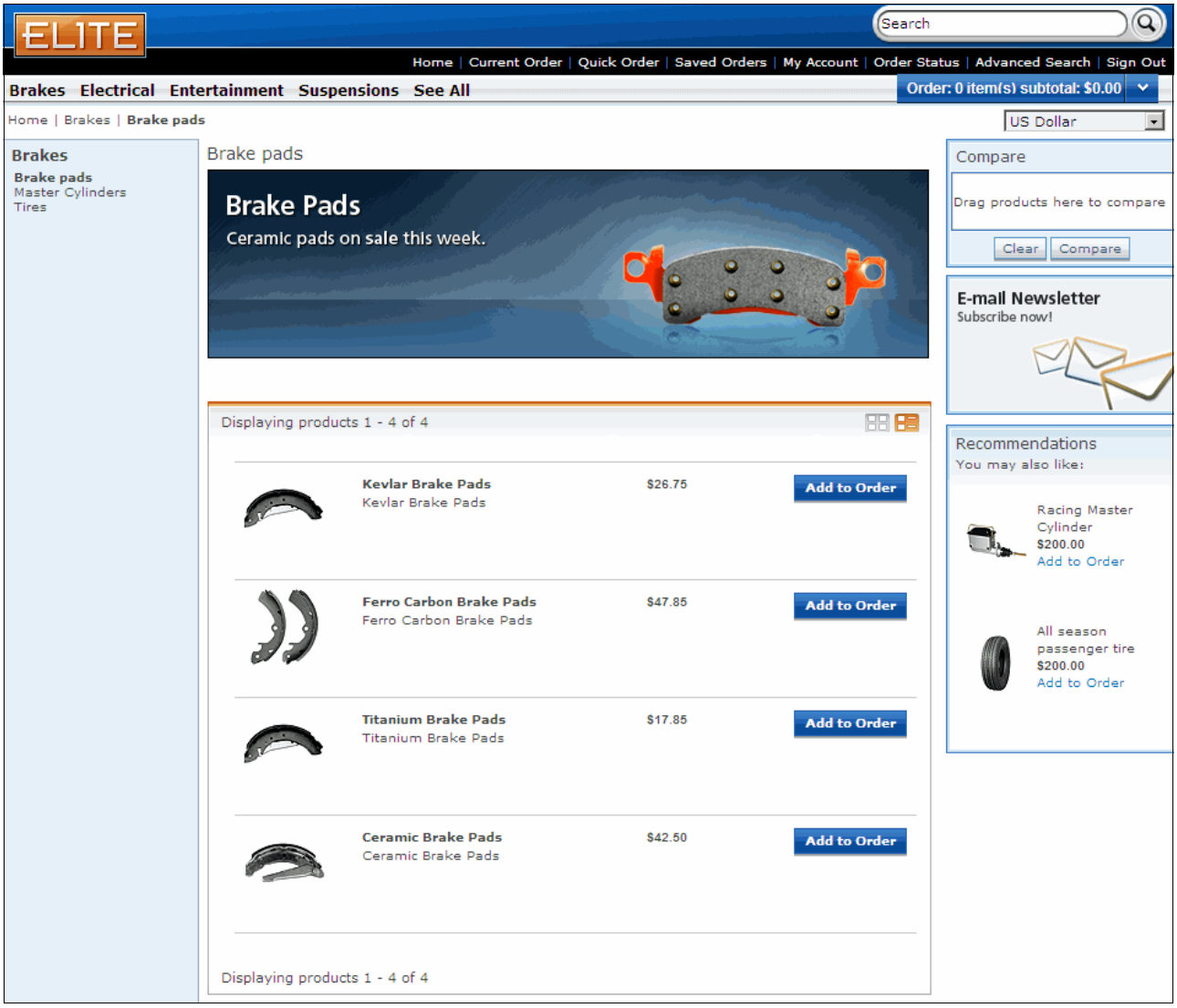


Figure 9-15 Storefront based on catalog filter CF2

If Nikit, who belongs to the ITSO Electrical systems organization (ITSO Electrical systems contract), logs in to the storefront, Nikit can see only Electrical category products, as shown in Figure 9-16.

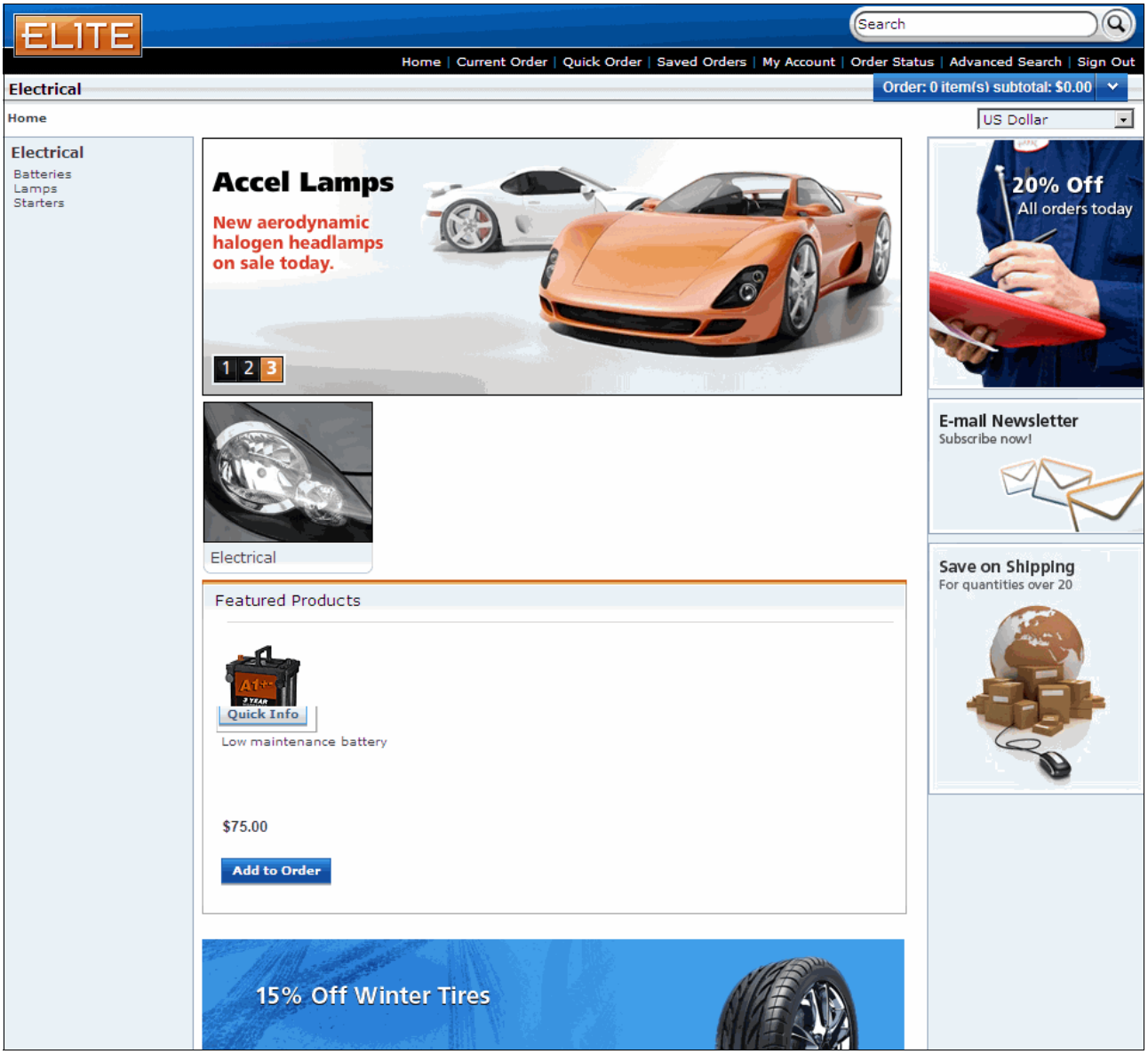


Figure 9-16 Storefront based on catalog filter CF3

Part 4

Appendixes

A

Log file lookup

Table A-1 maps an activity to its associated log file.

Table A-1 Log file reference

Activity	Log file
Installing Feature Pack 2	*WC_installdir*/logs/FEP2/*
Creating the Search Profile (running **ws-ant**)	*WAS_install_dir*/logs/manageprofiles/*
Enabling WebSphere Commerce foundation feature	*WC_installdir/*instances/*instance_name/*logs/enablefoundation_*timestamp*.log
Enabling StoreEnhancement feature	*WC_installdir/*instances/*instance_name/*logs/enablestore-enhancements_*timestamp*.log
`setupsearchIndex.sh` on WebSphere Commerce server	*WC_installdir*/components/foundation/subcomponents/search/log/wc-search-index-setup.log
`setupsearchIndex.sh` on remote search server	*working_dir*/search/log/wc-search-index-setup.log
`di-preprocess.sh`	*WC_installdir*/logs/wc-dataimport-preprocess.log
`di-buildindex.sh`	*WC_installdir*/logs/wc-dataimport-preprocess.log
Replication on Master and Slave machines	*WAS_installdir*/profiles/*search_profile_name*/logs/*search_server_name*/SystemOut.log
Solr server	*WAS_installdir*/profiles/*search_profile_name*/logs/*search_server_name*/SystemOut.log
Commerce server	*WAS_installdir*/profiles/*commerce_profile_name*/logs/server1/trace.log

Trace search: To get the trace search on the Commerce server, add the following trace string:

```
com.ibm.commerce.search.*=all: com.ibm.commerce.foundation.*=all
```

Back up files: Running `di-buildindex.sh` writes to the same log file as `di-preprocess.sh`. Running `di-buildindex.sh` after `di-preprocess.sh` overwrites the contents from `di-preprocess.sh`.

This behavior can be changed by increasing the number of `wc-dataimport-preprocess.logs` that are kept. Navigate to the directory named `WC_installdir/instances/<instancename>/xml/config/dataimport`, and open the file `logging.properties`. Find the line:

```
# Number of output files to cycle through
java.util.logging.FileHandler.count=1
```

And, increase the count value.

The `logging.properties` file is also where the logging detail can be changed.

Find the line:

```
# Default global logging level, INFO
.level=INFO
```

And, change `.level=INFO` to `.level=FINEST`

B

Configuration files

The following list shows the major configuration files that we have discussed:

- `wc-search.xml`

  This file is the major configuration file that is used to define and map interactions between the WebSphere Commerce and search servers. This file is located at:

  WC_ear_dir`/xml/config/com.ibm.commerce.catalog-fep/wc-search.xml`

- `wc-dataimport-preprocess-*.xml` files

  These files set up and populate the temporary tables that are used during `di-preprocess.sh`. There are multiple files used, for example, `wc-dataimport-preprocess-attribute-dictionary.xml` and `wc-dataimport-preprocess-common.xml`. All of these files are located at:

  WC_installdir`/instances/`*instance_name*`/search/pre-ProcessConfig/`*MC_catalog_ID*`/database_type/`

- `schema.xml`

  This file defines the fields that are used in the search index, and it sets up the field properties, for example, indexed, MultiValued, and so on. This file is located at:

  WC_installdir`/instances/`*instance_name*`/search/solr/home/MC_`*Icatalog_ID*`/`*language*`/CatalogEntry/conf/schema.xml`

- `wc-data-config.xml`

  This file is used to merge the temporary tables into a single index, and it maps the columns to field names. This file is located at:

  WC_installdir`instances/`*instance_name*`/search/solr/home/MC_`*Icatalog_ID*`/`*language*`/CatalogEntry/conf/wc-data-config.xml`

- `solrconfig.xml`

  This file is the Solr-specific configuration file. This file is used to set up replication, the default request handler, and so on. This file is located on the WebSphere Commerce and Search machines. On the WebSphere Commerce machine, this file is located at:

  WC_installdir`/instances/`*instance_name*`/search/solr/home/MC_`*Icatalog_ID*`/`*language*`/CatalogEntry/conf/solrconfig.xml`

On the Search machines, this file is located at:

solrhome/MC_*masterCatalogId*/*language*/CatalogEntry/conf/solrconfig.xml

- SRCHCONF table

  This table is used in `di-preprocess` and `di-buildindex` to point the Commerce server to the proper Indexing machine. This file is located in the commerce database.

- `solr-deploy.properties`

  This file contains the configuration information that is used when deploying the Solr application on a remote machine. This file is created during foundation enablement, and it is located at:

  WC_installdir/components/foundation/subcomponents/search/deploy/properties/solr-deploy.properties

C

Index design and data load

This appendix is a WebSphere Commerce V7.0 runtime version of Chapter 7, “Index design and data load” on page 179.

C.1 IBM WebSphere Commerce V7.0 catalog index SQL

The WebSphere Commerce V7.0 search uses the catalog index. We deployed our index locally with the standard configuration on SUSE 11 Enterprise.

C.1.1 Setting up the search index

Run the `setupSearchIndex.sh` to both deploy your catalog index structure to the WebSphere Commerce V7.0 server and configure WebSphere Commerce V7.0 to use the Solr index. This utility ensures that your index is built successfully using your WebSphere Commerce master catalog data. Follow these steps:

1. Start servers (see Example C-1):
   a. Log on as `reduser/passw0rd`.
   b. Open the Konsole terminal.

Example C-1 Start servers

```
redbooks@linux-8sjm:~>su - db2inst1
Password: passw0rd
db2inst1@linux-8sjm:~>
db2inst1@linux-8sjm:~>./sqllib/adm/db2start
db2inst1@linux-8sjm:~>exit
logout
redbooks@linux-8sjm:~>
redbooks@linux-8sjm:~>su
Password: passw0rd
linux-8sjm:/home/redbooks #
linux-8sjm:/home/redbooks #/opt/IBMIHS/bin/apachectl -k start -f
/opt/IBM/WebSphere/CommerceServer70/instances/demo/httpconf/httpd.conf
linux-8sjm:/home/redbooks #exit
logout
redbooks@linux-8sjm:~>
redbooks@linux-8sjm:~>/opt/IBM/WebSphere/AppServer/profiles/demo/bin/startServer.sh server1
redbooks@linux-8sjm:~>/opt/IBM/WebSphere/AppServer/profiles/demo_solr/bin/startServer.sh
solrServer
```

2. Log on as a WebSphere Commerce non-root user.
3. Change the directory:

   ```
   cd
   /opt/IBM/WebSphere/CommerceServer70/components/foundation/subcomponents/search/bin
   ```

4. Run `setupSearchIndex.sh`:

   ```
   ./setupSearchIndex.sh -instance demo -masterCatalogId 10001 -dbuser db2inst1
   -dbuserpwd passw0rd
   ```

5. Ensure that `setupSearchIndex.sh` ran successfully:
   a. Check the log file `wc-search-index-setup.log` for errors:
      i. Change the directory:

         ```
         cd
         /opt/IBM/WebSphere/CommerceServer70/components/foundation/subcomponents/search/log
         ```

      ii. View the log file:

         ```
         vi wc-search-index-setup.log
         ```

   b. Verify that the following assets were created (see Figure C-1):

      ```
      cd /opt/IBM/WebSphere/CommerceServer70/instances/demo/search/solr/home
      ```

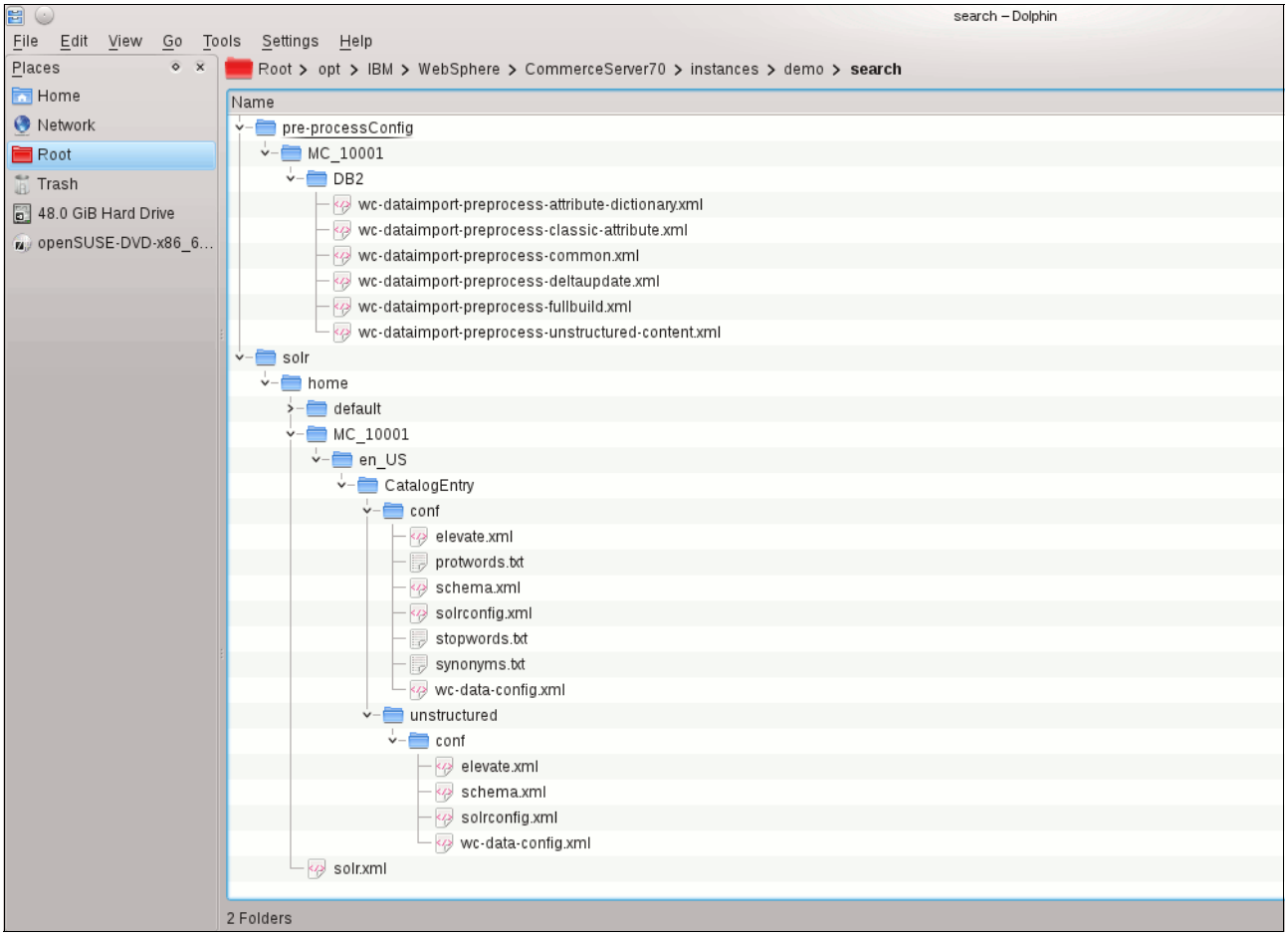


Figure C-1 Commerce search assets

Verify the following DB2 assets:

```
select * from SRCHATTR;
select * from SRCHATTRPROP;
select * from SRCHCONF;
select * from SRCHSTATS;
select * from SRCHTERM;
select * from SRCHTERMASSOCS;
```

6. Restart the WebSphere Commerce V7.0 search server.

Additional Information: For more information, go to this website:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchsetuplocal.htm

C.1.2 Preprocessing the index data

Preprocess the search index data to prepare your WebSphere Commerce V7.0 data for indexing. The preprocess utility, `di-preprocess.sh`, extracts, flattens, and imports your data into a set of temporary tables inside the WebSphere Commerce V7.0 database. This preprocessed data is then used by the index building utility, `di-buildindex.sh`, to create the indexes using Solr's Data Import Handler (DIH). The preprocessed data in the temporary tables is from the WebSphere Commerce V7.0 base schema.

Follow these steps:

1. Start the servers. See Example C-1 on page 256.
2. Log on as a WebSphere Commerce non-root user.
3. Change the directory:

   ```
   cd /opt/IBM/WebSphere/CommerceServer70/bin
   ```

4. Run `di-preprocess.sh`:

   ```
   ./di-preprocess.sh ../instances/demo/search/pre-processConfig/MC_10001/DB2
   -instance demo -dbuser db2inst1 -dbuserpwd passw0rd -fullbuild true -localename
   en_US
   ```

5. Ensure that `di-preprocess.sh` ran successfully:
   a. Change the directory:

      ```
      cd /opt/IBM/WebSphere/CommerceServer70/logs/
      ```

   b. View the log file:

      ```
      vi wc-dataimport-preprocess.log
      ```

   c. Verify the following DB2 changes (see Figure C-2).

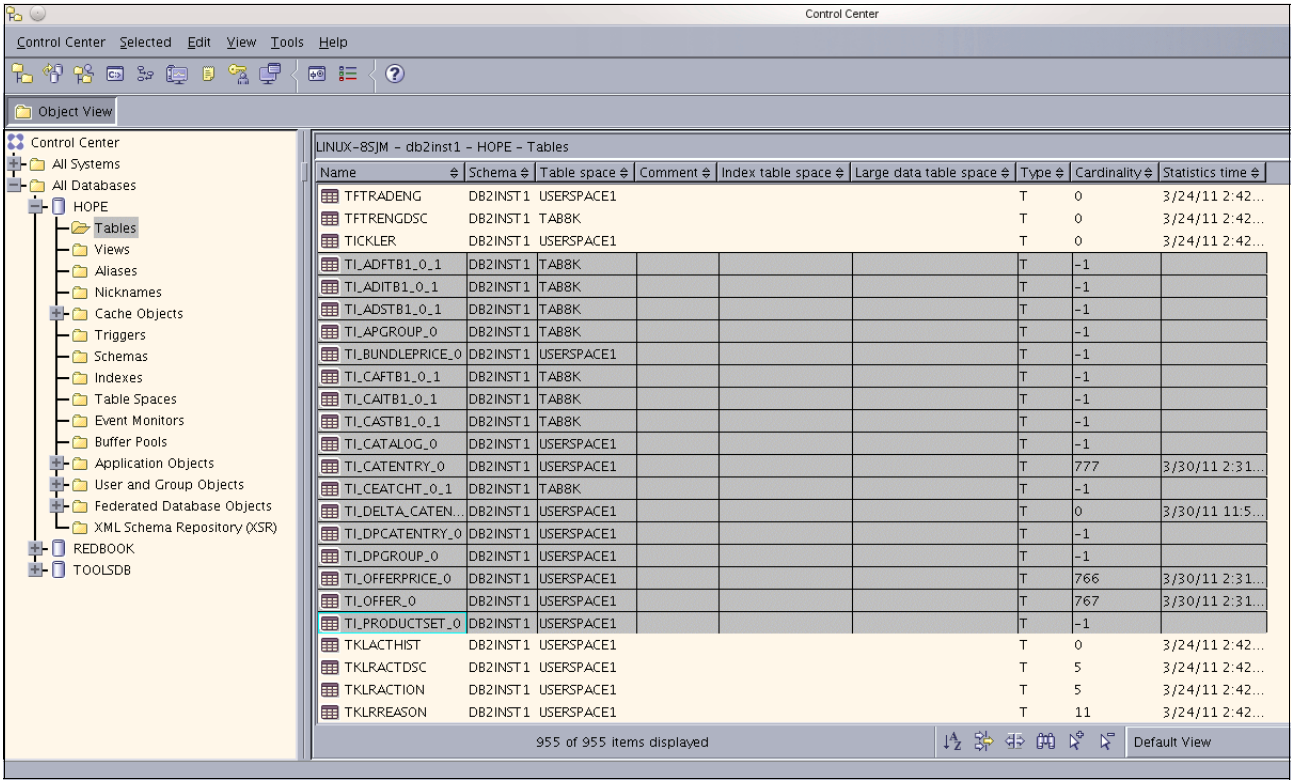


Figure C-2 Updated DB2

> **Additional information:** For more information, go to this website:
>
> http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchbuildpre.htm

C.1.3 Building the search index

You build the WebSphere Commerce search index using the index building utility, `di-buildindex.sh`. It is a wrapping utility that updates the information in the master index using Solr's Data Import Handler (DIH) service to build the index, either partially through delta index updates or completely through full index builds. When there are multiple indexes, for example, each language using its own separate index, the index is built multiple times. Follow these steps to build the index:

1. Start the servers (see Example C-1 on page 256).
2. Log on as a WebSphere Commerce non-root user.
3. Change the directory:

   ```
   cd /opt/IBM/WebSphere/CommerceServer70/bin
   ```

4. Run `di-buildindex.sh`:

   ```
   ./di-buildindex.sh -instance demo -masterCatalogId 10001 -dbuser db2inst1
   -dbuserpwd passw0rd -localename en_US -fullbuild true
   ```

   Important: Running `di-buildindex.sh` with a non-master `catalogId` raises this exception:

   ```
   com.ibm.commerce.foundation.dataimport.exception.DataImportApplicationExcept
   ion: An error occurred while setting up the search configuration.
   ```

5. Ensure that `di-buildindex.sh` ran successfully:

   a. Change the directory:

      ```
      cd /opt/IBM/WebSphere/CommerceServer70/logs
      ```

   b. View the log file (see Example C-2 and Figure 7-4 on page 184):

      ```
      vi wc-dataimport-preprocess.log
      ```

      Important:

      - The `di-buildindex.sh` utility overwrites the preprocess log file.
      - Before running `di-buildindex.sh`, back up `wc-dataimport-preprocess.log`.

Example C-2 wc-dataimport-preprocess.log

```
Mar 30, 2011 2:29:27 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain logStartDate
INFO: Data import pre-processing started:Wed Mar 30 14:29:27 EDT 2011
Mar 30, 2011 2:29:28 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain main
INFO: Data import pre-processing initialization completed in 1.034 seconds.
Mar 30, 2011 2:29:31 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:31 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:31 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:31 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:31 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 2.156 seconds for table TI_CATENTRY_0.
Mar 30, 2011 2:29:34 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 3.014 seconds for table TI_ADSTB1_0_#lang_tag#.
```

```
Mar 30, 2011 2:29:34 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 0.662 seconds for table TI_ADITB1_0_#lang_tag#.
Mar 30, 2011 2:29:35 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 0.88 seconds for table TI_ADFTB1_0_#lang_tag#.
Mar 30, 2011 2:29:36 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:36 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:36 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 0.77 seconds for table TI_CATALOG_0.
Mar 30, 2011 2:29:37 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:37 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:37 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
Mar 30, 2011 2:29:37 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 0.881 seconds for table TI_DPCATENTRY_0.
Mar 30, 2011 2:29:38 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:38 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:38 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 1.035 seconds for table TI_DPGROUP_0.
Mar 30, 2011 2:29:39 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 0.754 seconds for table TI_PRODUCTSET_0.
Mar 30, 2011 2:29:40 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:40 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:40 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
```

```
INFO: Data import pre-processing completed in 1.579 seconds for table TI_OFFER_0.
Mar 30, 2011 2:29:41 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:41 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:41 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 1.035 seconds for table TI_OFFERPRICE_0.
Mar 30, 2011 2:29:43 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:43 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:43 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 1.632 seconds for table TI_BUNDLEPRICE_0.
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor
process(DataProcessingConfig, Connection, boolean, String)
INFO: Process the master catalog...
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor populateTable
INFO: Beginning flush to database of 777 catalog entries...
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor populateTable
INFO: The batch is being executed...
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor populateTable
INFO: Batch execution complete.
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor populateTable
INFO: End flush to database.
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor
process(DataProcessingConfig, Connection, boolean, String)
INFO: Master catalog processing complete.
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor
process(DataProcessingConfig, Connection, boolean, String)
INFO: Process the sales catalogs...
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor populateTable
INFO: Beginning flush to database of 34 catalog entries...
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor populateTable
INFO: The batch is being executed...
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor populateTable
```

```
INFO: Batch execution complete.
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor populateTable
INFO: End flush to database.
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor
process(DataProcessingConfig, Connection, boolean, String)
INFO: Sales catalog processing complete.
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.CatalogHierarchyDataPreProcessor
process(DataProcessingConfig, Connection, boolean, String)
INFO:
======================================================================
Catalog hierarchy search index data pre-processor
======================================================================
Batch size: 10000
Total catalog entries that have had their hierarchy determined: 811
======================================================================
Mar 30, 2011 2:29:45 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 2.264 seconds for table TI_APGROUP_0.
Mar 30, 2011 2:29:47 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: The batch is being executed...
Mar 30, 2011 2:29:47 PM
com.ibm.commerce.foundation.dataimport.preprocess.AbstractDataPreProcessor
populateTable(PreparedStatement, Long, Connection)
INFO: Batch execution completed.
Mar 30, 2011 2:29:47 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 1.305 seconds for table TI_CEATCHT_0_#lang_tag#.
Mar 30, 2011 2:29:49 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 2.016 seconds for table TI_CASTB1_0_#lang_tag#.
Mar 30, 2011 2:29:49 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 0.681 seconds for table TI_CAITB1_0_#lang_tag#.
Mar 30, 2011 2:29:50 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain processDataConfig
INFO: Data import pre-processing completed in 0.655 seconds for table TI_CAFTB1_0_#lang_tag#.
Mar 30, 2011 2:29:50 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain logExitCode
INFO:
Program exiting with exit code: 0.
Data import pre-processing completed successfully with no errors.
Mar 30, 2011 2:29:50 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain logEndDateAndTime
INFO: Data import pre-processing ended:Wed Mar 30 14:29:50 EDT 2011
Mar 30, 2011 2:29:50 PM
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain logEndDateAndTime
INFO: Data import pre-processing completed in 23.128 seconds.
```

c. Verify that the `MC_10001_CatalogEntry_en_US` index is on the disk (see Figure C-3).

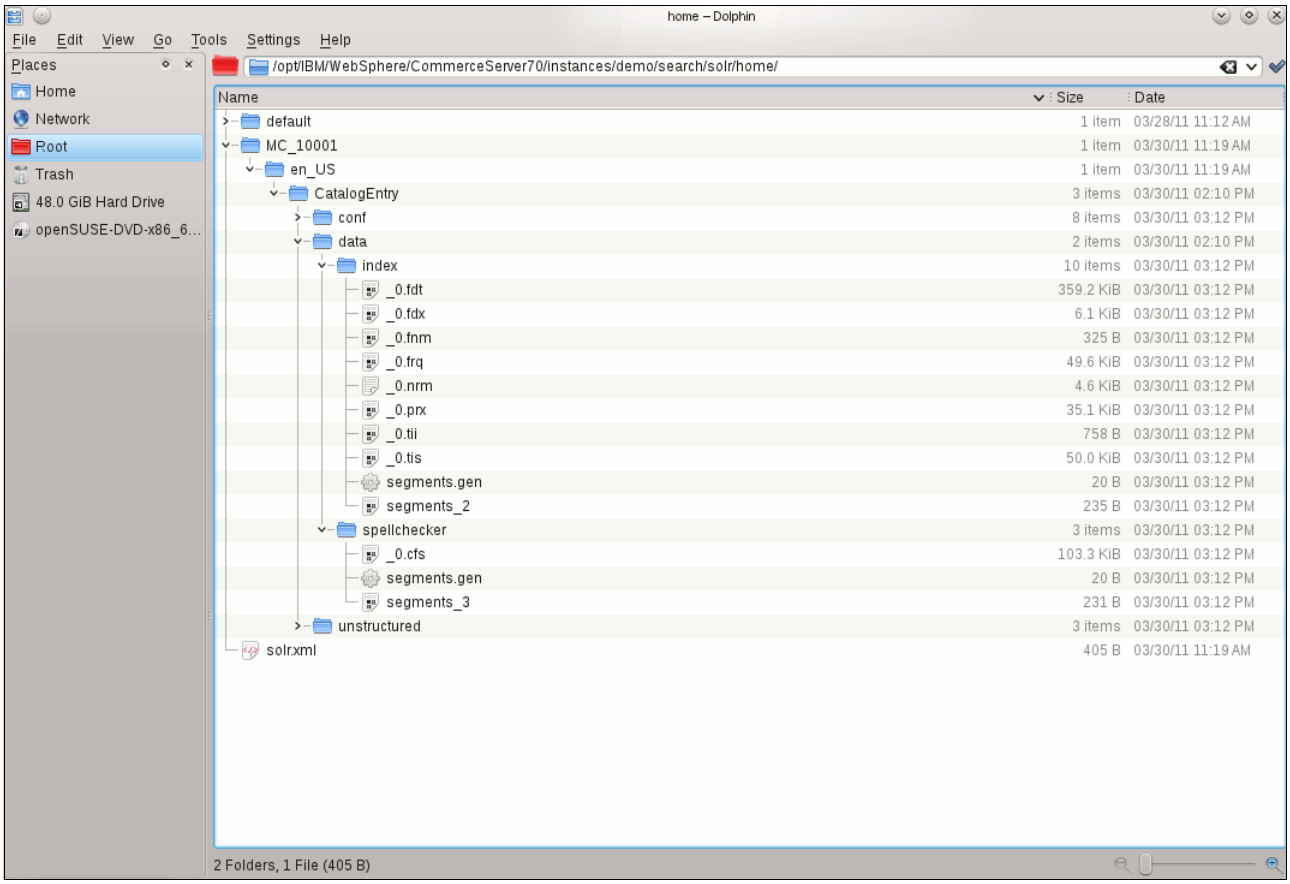


Figure C-3 MC_10001_CatalogEntry_en_US index assets

d. Check the `MC_10001_CatalogEntry_en_US` index status (see Example C-3):

```
firefox
http://linux-8sjm.site:3737/solr/admin/cores?action=status&core=MC_10001_Cat
alogEntry_en_US
```

Example C-3 Status response

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">10</int>
  </lst>
  <lst name="status">
    <lst name="MC_10001_CatalogEntry_en_US">
      <str name="name">MC_10001_CatalogEntry_en_US</str>
      <str name="instanceDir">

/opt/IBM/WebSphere/CommerceServer70/instances/demo/search/solr/home/MC_10001/en_US/CatalogEntry/
      </str>
      <str name="dataDir">

/opt/IBM/WebSphere/CommerceServer70/instances/demo/search/solr/home/MC_10001/en_US/CatalogEntry/
data/
      </str>
```

```
      <date name="startTime">2011-03-31T20:27:20.149Z</date>
      <long name="uptime">43528016</long>
      <lst name="index">
        <int name="numDocs">777</int>
        <int name="maxDoc">777</int>
        <long name="version">1301508646147</long>
        <bool name="optimized">true</bool>
        <bool name="current">true</bool>
        <bool name="hasDeletions">false</bool>
        <str name="directory">

org.apache.lucene.store.NIOFSDirectory:org.apache.lucene.store.NIOFSDirectory@/opt/IBM/WebSphere
/CommerceServer70/instances/demo/search/solr/home/MC_10001/en_US/CatalogEntry/data/index
        </str>
        <date name="lastModified">2011-03-30T19:12:43Z</date>
      </lst>
    </lst>
  </lst>
</response>
```

e. Query the `MC_10001_CatalogEntry_en_US` index (see Example C-4):

```
http://linux-8sjm.site:3737/solr/MC_10001_CatalogEntry_en_US/select/?q=red&fl=*
```

Example C-4 Query response

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">38</int>
    <lst name="params">
      <str name="fl">*</str>
      <str name="q">red</str>
    </lst>
  </lst>
  <result name="response" numFound="30" start="0">
    <doc>
      <int name="buyable">1</int>
      <arr name="catalog_id">
        <long>10001</long>
      </arr>
      <long name="catentry_id">10727</long>
      <str name="catenttype_id_ntk_cs">ProductBean</str>
      <str name="fullImage">
        images/catalog/grocery_health/grocery_health_160x160/Red_Cabbage.jpg
      </str>
      <long name="member_id">7000000000000000002</long>
      <str name="mfName">MadisonsGrocer</str>
      <str name="mfName_ntk">MadisonsGrocer</str>
      <str name="mfName_ntk_cs">MadisonsGrocer</str>
      <str name="name">Red Cabbage</str>
      <arr name="parentCatgroup_id_facet">
        <str>10001_10053</str>
      </arr>
     <arr name="parentCatgroup_id_search">
        <str>10001_10053</str>
        <str>10001_10052</str>
```

```
      </arr>
      <str name="partNumber_ntk">MadisonsGCR-009</str>
      <float name="price_USD">2.99</float>
      <int name="published">1</int>
      <str name="shortDescription">Whole red cabbages.</str>
      <int name="storeent_id">10001</int>
      <str name="thumbnail">
        images/catalog/grocery_health/grocery_health_70x70/Red_Cabbage.jpg
      </str>
    </doc>
    <!-- 29 documnents removed -->
  <result>
</response>
```

Additional information: For more information, go to this website:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchbuildindex.htm

C.1.4 Maintenance

Over time, the WebSphere Commerce V7.0 search index becomes out of sync with the latest production data. To maintain the synchronization of the search result data, you must perform re-indexing during normal business operations.

Additional information: For more information, go to this website:

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.developer.doc/concepts/csdsearchcontentstructureindex.htm

C.2 Scenario: Display only products with inventory

A common business requirement is to only display products on the storefront that have inventory. When using WebSphere Commerce V7.0 search, this capability requires performing an index update. There is always a period of time, *t1*, when products in the catalog index potentially have no inventory. Our implementation minimizes this time period and gives the shopper a better shopping experience. Follow this sequence of steps:

1. The order is placed for product *X*.
2. The order is fulfilled for product *X*. Product *X* now has 0 inventory.
3. Start *t1*.
4. Perform the delta re-indexing:
   a. For each product that has 0 inventory, generate and execute a `ChangeCatalogEntry` event.
   b. Generate and execute an index synchronization event.
5. Stop *t1*.

C.2.1 Adding the inventory field to the catalog index

For this solution, we customize the WebSphere Commerce V7.0 index by adding an additional field for inventory. We must flatten out the catalog entry/inventory data, which involves creating a custom preprocessor and custom temporary index tables.

Customizing: Your table name must start with an *X* to avoid conflict with the default WebSphere Commerce tables, for example, `XI_INVENTORY_0`.

Follow these steps to add the inventory field to the catalog index:

1. Configure the search preprocessor:
   a. Create a new custom preprocess configuration file:
      i. Launch **Kickoff** → **Applications** → **Utilities** → **Editor** → **KWrite**.
      ii. Click **File** → **Save As**:

      `/opt/IBM/WebSphere/CommerceServer70/instances/demo/search/pre-processConfig/MC_10001/DB2/wc-dataimport-preprocess-custom-inventory.xml`

      iii. Add our code, which is shown in Example C-5.

Example C-5 wc-dataimport-preprocess-custom-inventory.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<_config:DIHPreProcessConfig
    xmlns:_config="http://www.ibm.com/xmlns/prod/commerce/foundation/config"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ibm.com/xmlns/prod/commerce/foundation/config ../../xsd/wc-dataimport-preprocess.xsd">


<!-- one level PRODUCT_ITEM catentry parent -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
    <_config:table definition="CREATE TABLE XI_DPCATENTRY_PRODUCT_ITEM_0 (CATENTRY_ID BIGINT NOT NULL, CATENTRY_ID_PARENT BIGINT, PRIMARY
KEY (CATENTRY_ID))" name="XI_DPCATENTRY_PRODUCT_ITEM_0"/>
    <_config:query sql="SELECT CATENTRY_ID_CHILD, CATENTRY_ID_PARENT FROM CATENTREL R, TI_CATENTRY_0 C WHERE R.CATENTRY_ID_CHILD =
C.CATENTRY_ID AND R.CATRELTYPE_ID='PRODUCT_ITEM' ORDER BY CATENTRY_ID_CHILD"/>
    <_config:mapping>
        <_config:key queryColumn="CATENTRY_ID_CHILD" tableColumn="CATENTRY_ID"/>
        <_config:column-mapping>
            <_config:column-column-mapping>
                <_config:column-column queryColumn="CATENTRY_ID_PARENT" tableColumn="CATENTRY_ID_PARENT" />
            </_config:column-column-mapping>
        </_config:column-mapping>
    </_config:mapping>
</_config:data-processing-config>

<!-- one level BUNDLE_COMPONENT catentry parent -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
    <_config:table definition="CREATE TABLE XI_DPCATENTRY_BUNDLE_COMPONENT_0 (CATENTRY_ID BIGINT NOT NULL, CATENTRY_ID_PARENT BIGINT,
PRIMARY KEY (CATENTRY_ID))" name="XI_DPCATENTRY_BUNDLE_COMPONENT_0"/>
    <_config:query sql="SELECT CATENTRY_ID_CHILD, CATENTRY_ID_PARENT FROM CATENTREL R, TI_CATENTRY_0 C WHERE R.CATENTRY_ID_CHILD =
C.CATENTRY_ID AND R.CATRELTYPE_ID='BUNDLE_COMPONENT' ORDER BY CATENTRY_ID_CHILD"/>
    <_config:mapping>
        <_config:key queryColumn="CATENTRY_ID_CHILD" tableColumn="CATENTRY_ID"/>
        <_config:column-mapping>
            <_config:column-column-mapping>
                <_config:column-column queryColumn="CATENTRY_ID_PARENT" tableColumn="CATENTRY_ID_PARENT" />
            </_config:column-column-mapping>
        </_config:column-mapping>
    </_config:mapping>
</_config:data-processing-config>

<!-- ItemBean inventory -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
  <!-- leaving ffmcenterId out as any inventory is inventory -->
  <_config:table definition="CREATE TABLE XI_ITEM_INVENTORY_0 (CATENTRY_ID BIGINT NOT NULL, STORE_ID BIGINT NOT NULL, QUANTITY BIGINT,
PRIMARY KEY (CATENTRY_ID,STORE_ID))" name="XI_ITEM_INVENTORY_0"/>
```

```
  <_config:query sql="SELECT I.CATENTRY_ID, I.STORE_ID, SUM( CAST(I.QUANTITY as BIGINT) ) quantity FROM INVENTORY I,
XI_DPCATENTRY_PRODUCT_ITEM_0 CE WHERE I.STORE_ID IN ( SELECT STOREENT_ID from STORECAT WHERE CATALOG_ID=10001 ) AND I.CATENTRY_ID =
CE.CATENTRY_ID GROUP BY I.CATENTRY_ID, I.STORE_ID ORDER BY I.CATENTRY_ID"/>
  <_config:mapping>
    <_config:key queryColumn="CATENTRY_ID" tableColumn="CATENTRY_ID"/>
    <_config:column-mapping>
      <_config:column-column-mapping>
    <_config:column-column queryColumn="STORE_ID" tableColumn="STORE_ID" />
    <_config:column-column queryColumn="QUANTITY" tableColumn="QUANTITY" />
      </_config:column-column-mapping>
    </_config:column-mapping>
  </_config:mapping>
</_config:data-processing-config>

<!-- ProductBean inventory: this is the inventory of the ItemBean(s) with a PRODUCT_ITEM  relationship -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
  <!-- leaving ffmcenterId out as any inventory is inventory -->
  <_config:table definition="CREATE TABLE XI_PRODUCT_INVENTORY_0 (CATENTRY_ID BIGINT NOT NULL, CATENTRY_ID_PARENT VARCHAR(512), QUANTITY
BIGINT, PRIMARY KEY (CATENTRY_ID))" name="XI_PRODUCT_INVENTORY_0"/>
  <_config:query sql="SELECT T.CATENTRY_ID, T.CATENTRY_ID_PARENT, X.QUANTITY FROM XI_ITEM_INVENTORY_0 X INNER JOIN
XI_DPCATENTRY_PRODUCT_ITEM_0 T ON X.CATENTRY_ID = T.CATENTRY_ID"/>
  <_config:mapping>
    <_config:key queryColumn="CATENTRY_ID" tableColumn="CATENTRY_ID"/>
    <_config:column-mapping>
      <_config:column-column-mapping>
    <_config:column-column queryColumn="CATENTRY_ID_PARENT" tableColumn="CATENTRY_ID_PARENT" />
    <_config:column-column queryColumn="QUANTITY" tableColumn="QUANTITY" />
      </_config:column-column-mapping>
    </_config:column-mapping>
  </_config:mapping>
</_config:data-processing-config>

<!-- BundleBean inventory: this is the inventory of the ItemBean(s) with a BUNDLE_COMPONENT  relationship -->
<_config:data-processing-config processor="com.ibm.commerce.foundation.dataimport.preprocess.StaticAttributeDataPreProcessor"
batchSize="500">
  <!-- leaving ffmcenterId out as any inventory is inventory -->
  <_config:table definition="CREATE TABLE XI_BUNDLE_INVENTORY_0 (CATENTRY_ID BIGINT NOT NULL, CATENTRY_ID_PARENT VARCHAR(512), QUANTITY
BIGINT, PRIMARY KEY (CATENTRY_ID))" name="XI_BUNDLE_INVENTORY_0"/>
  <_config:query sql="SELECT T.CATENTRY_ID, T.CATENTRY_ID_PARENT, X.QUANTITY FROM XI_ITEM_INVENTORY_0 X INNER JOIN
XI_DPCATENTRY_BUNDLE_COMPONENT_0 T ON X.CATENTRY_ID = T.CATENTRY_ID"/>
  <_config:mapping>
    <_config:key queryColumn="CATENTRY_ID" tableColumn="CATENTRY_ID"/>
    <_config:column-mapping>
      <_config:column-column-mapping>
    <_config:column-column queryColumn="CATENTRY_ID_PARENT" tableColumn="CATENTRY_ID_PARENT" />
    <_config:column-column queryColumn="QUANTITY" tableColumn="QUANTITY" />
      </_config:column-column-mapping>
    </_config:column-mapping>
  </_config:mapping>
</_config:data-processing-config>
</_config:DIHPreProcessConfig>
```

b. Preprocess the search index data. See C.1.2, “Preprocessing the index data” on page 258.

c. Verify the custom index tables with SQL:

```
select * from XI_DPCATENTRY_PRODUCT_ITEM_0;
select * from XI_DPCATENTRY_BUNDLE_COMPONENT_0;
select * from XI_ITEM_INVENTORY_0;
select * from XI_PRODUCT_INVENTORY_0;
select * from XI_BUNDLE_INVENTORY_0;
```

Troubleshooting: The following exception sometimes occurs when running the `di-preprocess.sh` utility while any of the TI_*XXXX* or XI_*XXXX* tables are in use by another application, for example, a query in the DB2 V9.7 Control Center:

```
com.ibm.commerce.foundation.dataimport.preprocess.DataImportPreProcessorMain
handleExecutionException
SEVERE: CWFDIH0002: An SQL exception was caught. The following error occurred:
DB2 SQL Error: SQLCODE=-601, SQLSTATE=42710,
SQLERRMC=DB2INST1.<TABLE_NAME>;TABLE, DRIVER=4.7.85
```

2. Configure the Data Import Handler mapping:
   a. Launch **Kickoff** → **Applications** → **Utilities** → **Editor** → **KWrite**.
   b. Click **File** → **Open**:

      ```
      /opt/IBM/WebSphere/CommerceServer70/instances/demo/search/solr/home/MC_10001
      /en_US/CatalogEntry/conf/schema.xml
      ```

   c. Locate `<field name="defaultSearch" type="wc_text" indexed="true" stored="false" multiValued="true"/>` and insert our code, which is shown in Example C-6.

Example C-6 Inventory field definition

```
<!-- Inventory's quantity: map to table: INVENTORY -->
  <field name="xquantity" type="long" indexed="true" stored="true" multiValued="false" />
```

   d. Save your changes.
   e. Launch → **Kickoff** → **Applications** → **Utilities** → **Editor** → **KWrite**.
   f. Click **File** → **Open**:

      ```
      /opt/IBM/WebSphere/CommerceServer70/instances/demo/search/solr/home/MC_10001
      /en_US/CatalogEntry/conf/wc-data-config.xml
      ```

   g. Add the following columns and tables in Example C-7 to the select SQL.
   h. Save and close the file.
   i. Restart the WebSphere Commerce V7.0 search server.
   j. Build the search index. See C.1.3, “Building the search index” on page 259.
   k. Verify the data:

      ```
      firefox
      http://linux-8sjm.site:3737/solr/MC_10001_CatalogEntry_en_US/select/?q=catentty
      pe_id_ntk_cs:ItemBean&fl=*
      ```

Troubleshoot: If your `XI_<TABLENAME>` is missing data in a column, check that you have all of your column/column mappings.

Example C-7 wc-data-config.xml

```
<!-- WebSphere Commerce Solr Data Import Handler configuration -->
<dataConfig>
<dataSource name="WC database"
        type="JdbcDataSource"
        jndiName="jdbc/WebSphere Commerce Search DB2 DataSource demo"
        readOnly="true"
        autoCommit="true"
```

```
        transactionIsolation="TRANSACTION_READ_COMMITTED"
        holdability="CLOSE_CURSORS_AT_COMMIT"/>
<dataSource name="unstructuretmpfile"
        type="FileDataSource"
        basePath="/opt/IBM/WebSphere/CommerceServer70/instances/demo/search/solr/home/MC_10001/en_US/CatalogEntry/unstructured/temp/"/>
<document name="CatalogEntry">
<entity name="Product"
    dataSource="WC database"
    transformer="ClobTransformer, RegexTransformer"
    query="SELECT CATENTRY.CATENTRY_ID,CATENTRY.MEMBER_ID,CATENTRY.CATENTTYPE_ID,CATENTRY.PARTNUMBER,CATENTRY.MFPARTNUMBER,
        CATENTRY.MFNAME,CATENTRY.BUYABLE,STORECENT.STOREENT_ID,CATENTDESC.NAME,CATENTDESC.SHORTDESCRIPTION,
            ...
        TI_CATALOG.CATALOG PARENT_CATALOG_ID,
XI_PRODUCT_INVENTORY.QUANTITY XQUANTITY,
        TI_CASTB1.CAS_F1 CAS_F1ATTR, TI_CASTB1.CAS_F2 CAS_F2ATTR, TI_CASTB1.CAS_F3 CAS_F3ATTR,TI_CASTB1.CAS_F4
CAS_F4ATTR,TI_CASTB1.CAS_F5 CAS_F5ATTR,
            ...
        TI_ADFTB1.ADF_F1, TI_ADFTB1.ADF_F2, TI_ADFTB1.ADF_F3, TI_ADFTB1.ADF_F4, TI_ADFTB1.ADF_F5, TI_ADFTB1.ADF_F6, TI_ADFTB1.ADF_F7,
TI_ADFTB1.ADF_F8, TI_ADFTB1.ADF_F9, TI_ADFTB1.ADF_F10
        FROM CATENTRY
            INNER JOIN TI_CATENTRY_O TI_CATENTRY ON (CATENTRY.CATENTRY_ID=TI_CATENTRY.CATENTRY_ID)
            LEFT OUTER JOIN STORECENT ON (CATENTRY.CATENTRY_ID=STORECENT.CATENTRY_ID)
                ...
            LEFT OUTER JOIN TI_CATALOG_O TI_CATALOG ON (CATENTRY.CATENTRY_ID=TI_CATALOG.CATENTRY_ID)
LEFT OUTER JOIN XI_PRODUCT_INVENTORY_O XI_PRODUCT_INVENTORY ON (CATENTRY.CATENTRY_ID=
CAST(XI_PRODUCT_INVENTORY.CATENTRY_ID_PARENT AS BIGINT) )
                ...
            LEFT OUTER JOIN TI_ADFTB1_O_1 TI_ADFTB1 ON (CATENTRY.CATENTRY_ID=TI_ADFTB1.CATENTRY_ID)
        WHERE  CATENTRY.CATENTTYPE_ID='ProductBean'"

deltaImportQuery="SELECT CATENTRY.CATENTRY_ID,CATENTRY.MEMBER_ID,CATENTRY.CATENTTYPE_ID,CATENTRY.PARTNUMBER,CATENTRY.MFPARTNUMBER,
                ...
            TI_CATALOG.CATALOG PARENT_CATALOG_ID,
XI_PRODUCT_INVENTORY.QUANTITY XQUANTITY,
            TI_CASTB1.CAS_F1 CAS_F1ATTR, TI_CASTB1.CAS_F2 CAS_F2ATTR, TI_CASTB1.CAS_F3 CAS_F3ATTR,TI_CASTB1.CAS_F4
                ...
            TI_ADFTB1.ADF_F1, TI_ADFTB1.ADF_F2, TI_ADFTB1.ADF_F3, TI_ADFTB1.ADF_F4, TI_ADFTB1.ADF_F5, TI_ADFTB1.ADF_F6, TI_ADFTB1.ADF_F7,
TI_ADFTB1.ADF_F8, TI_ADFTB1.ADF_F9, TI_ADFTB1.ADF_F10
        FROM CATENTRY
            INNER JOIN TI_CATENTRY_O TI_CATENTRY ON (CATENTRY.CATENTRY_ID=TI_CATENTRY.CATENTRY_ID)
            LEFT OUTER JOIN STORECENT ON (CATENTRY.CATENTRY_ID=STORECENT.CATENTRY_ID)
                ...
            LEFT OUTER JOIN TI_CATALOG_O TI_CATALOG ON (CATENTRY.CATENTRY_ID=TI_CATALOG.CATENTRY_ID)
LEFT OUTER JOIN XI_PRODUCT_INVENTORY_O XI_PRODUCT_INVENTORY ON (CATENTRY.CATENTRY_ID=
CAST(XI_PRODUCT_INVENTORY.CATENTRY_ID_PARENT AS BIGINT) )
                ...
            LEFT OUTER JOIN TI_ADFTB1_O_1 TI_ADFTB1 ON (CATENTRY.CATENTRY_ID=TI_ADFTB1.CATENTRY_ID)
          WHERE  CATENTRY.CATENTTYPE_ID='ProductBean'"
deltaQuery="SELECT CATENTRY_ID FROM TI_CATENTRY_O WHERE CATENTTYPE_ID='ProductBean'"
deletedPkQuery="SELECT TI_D_CATENTRY_O.CATENTRY_ID FROM TI_D_CATENTRY_O, CATENTRY WHERE TI_D_CATENTRY_O.CATENTRY_ID=CATENTRY.CATENTRY_ID
AND CATENTRY.CATENTTYPE_ID = 'ProductBean'">
    <field column="CATENTRY_ID" name="catentry_id" />
        ...
    <field column="PRICE_GBP" name="price_GBP"/>
<field column="XQUANTITY" name="xquantity"/>
    <field column="CAS_F1ATTR"  clob="true"/>
        ...
</entity><!-- end Product-->
```

3. Add our quantity filter query to all queries:
   a. Launch **Kickoff** → **Applications** → **Utilities** → **Editor** → **KWrite**.
   b. Click **File** → **Open**:

      `/opt/IBM/WebSphere/CommerceServer70/instances/demo/search/solr/home/MC_10001/en_US/CatalogEntry/conf/solrconfig.xml`

   c. Locate `<requestHandler name="standard"...` and insert our code in Example C-8 on page 271.
   d. Save and close the file.
   e. Restart the WebSphere Commerce V7.0 search server.

f. Verify the quantity filter. This operation is successful if 744 documents are returned:

```
firefox
http://linux-8sjm.site:3737/solr/MC_10001_CatalogEntry_en_US/select/?q=*&fl=*
```

Example C-8 solrconfig.xml

```
<!-- WebSphere Commerce default search request handler -->
<requestHandler name="standard" class="solr.StandardRequestHandler" default="true">
   <!-- default values for query parameters -->
   <lst name="defaults">
      <str name="echoParams">explicit</str>
      <!--
      <int name="rows">10</int>
      <str name="fl">*</str>
      <str name="version">2.1</str>
      -->
   </lst>

<!-- Filter all queries to ensure returned catalog entries have quantity -->
   <lst name="appends">
      <str name="fq">xquantity:[1 TO *]</str>
   </lst>

   <arr name="last-components">
      <str>wc_spellcheck</str>
   </arr>
</requestHandler>
```

C.2.2 Index maintenance

See Chapter 7, “Index design and data load” on page 179.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

The following IBM Redbooks publications provide additional information about the topic in this document. This publication might be available in softcopy only.

- *Selling and Fulfillment Solutions using WebSphere Commerce and Sterling Order Management, SG24-7930-00*

You can search for, view, or download IBM Redbooks, Redpapers, webdocs, draft publications and additional materials, as well as order hardcopy IBM Redbooks publications, at this Web site:

ibm.com/redbooks

Online resources

These Web sites are also relevant as further information sources:

- WebSphere Commerce search

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.developer.doc/concepts/csdsearch.htm

- Opening the Management Center

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.management-center.doc/tasks/ttflogon.htm

- Changing your preferences

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.management-center.doc/tasks/ttfupdatepreference.htm

- Enabling the Management Center marketing features

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.management-center.doc/tasks/tsbenable_dup.htm

- Managing WebSphere Commerce search: Populating and building the search index data

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdmanagesearchtop.htm

- Common business tasks and their effect on the WebSphere Commerce search index

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/refs/rsdsearchindexhints.htm

- Index synchronization and delta updates in WebSphere Commerce search

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdsearchcontentstructureindex.htm

- Replicating WebSphere Commerce search

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdreplicatesearch.htm
- Setting up WebSphere Commerce search in a clustered environment

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdfederatesearch.htm
- Luke - Lucene Index Toolbox

  http://code.google.com/p/luke/downloads/list
- Cache invalidation

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.admin.doc/concepts/cdc_cacheinv.htm
- Apache Solr

  http://lucene.apache.org/solr/#intro
- Administering WebSphere Commerce search

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdmanagesearch.htm
- Preparing the operating system for product installation

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.installation.nd.doc/info/ae/ae/tins_prepare.html
- 7.0.0.15: WebSphere Application Server V7.0 Fix Pack 15 for Linux

  http://www-01.ibm.com/support/docview.wss?uid=swg24029073
- WebSphere Commerce Update Installer Version 6.1 and 7.0

  http://www-01.ibm.com/support/docview.wss?uid=swg24013502
- Creating a WebSphere Commerce instance

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.install.doc/tasks/tig_part_create_instances.htm
- Setting up the WebSphere Commerce search index structure for a specific master catalog locally

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchsetuplocal.htm
- Preprocessing the WebSphere Commerce search index data

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchbuildpre.htm
- Building the WebSphere Commerce search index

  http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/tasks/tsdsearchbuildindex.htm

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

WebSphere Commerce V7.0 Feature Pack 2 Search Solution Overview and Deployment

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

WebSphere Commerce V7.0 Feature Pack 2 Search Solution Overview and Deployment

Search results by using a new set of search tools

Work with B2C and B2B scenarios

Understand search using index life-cycle management

This IBM Redbooks publication focuses on the enhanced search capabilities of the newest release of WebSphere Commerce V7.0 Feature Pack 2. We divided this book into three parts to highlight search business-to-consumer (B2C) and search business-to-business (B2B) scenarios and a conceptual overview of the search solution.

This book can help you to enable the search features of WebSphere Commerce V7.0 Feature Pack 2 and set up the WebSphere Commerce search environment, experience the design and management of Solr indexes, and experience several B2C scenarios using the Madisons Feature Pack 2 (FEP2) store.

This book gives you a broad understanding of the WebSphere Commerce integrated search solution that provides control over search results through a new set of search management tools:

- Search term association in the Catalog tool
- Marketing web activity tool
- Marketing dialog activity tool
- Search activity tool

This book describes search index life-cycle management and examines each area, such as index building, synchronization, configuration, and replication. This book describes the entitlement feature that is provided through the WebSphere Commerce search framework. We also discuss a B2B scenario and provide the implementation using catalog filters and contracts.

This book is designed for use by WebSphere Commerce developers, practitioners, and solution architects in various industries.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7950-00 ISBN 0738435759